Trabajo de Final de Grado

Grado de Ingeniería en Tecnologías Industriales

Aplicación de técnicas de minería de datos para predecir la popularidad de noticias en Internet

MEMORIA

Autor: Joel Medina Jiménez
Director: Luis José Talavera Méndez

Convocatoria: Enero 2020

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Resumen

La minería de datos es un estudio cuyo uso se ha empezado ha desarrollar hace pocos años. Hay una gran cantidad de información, pero gran parte de esta es bastante compleja.

El presente trabajo se propone predecir, mediante la minería de datos, cuál será el nivel de popularidad de las publicaciones de una página web. Para llevarlo a cabo, se ha utilizado una metodología muy conocida para un buen uso de la minería de datos llamada *CRISP-DM.* Esta metodología se compone de diferentes etapas las cuales se han seguido en este proyecto, poniendo más énfasis en las últimas fases.

Para poder realizar este proyecto se ha usado un lenguaje de programación aprendido en el grado, llamado Python. Por medio del uso de librerías *Pandas y Sklearn* incluidas en Python, se ha conseguido crear modelos predictivos. También se ha empleado una plataforma llamada *Anaconda* donde mediante *Spyder,* se ha tratado de realizar una programación en Python más dinámica y así poder observar los resultados rápidamente.

ÍNDICE

1. Glosario

Palabras no vacías→ se entiende palabra no vacía como nombres, adjetivos, verbos y adverbios.

Palabras únicas→ son palabras técnicas que dan complejidad al texto.

Palabras clave→ se entienden como aquellas palabras que aportan una información importante y significativa acerca de un contenido.

Dataframe→ se entiende como una hoja de datos, donde cada fila corresponde a un objeto y cada columna a una variable.

2. Introducción

Estamos en una época en la que es habitual observar a personas que poseen un objeto electrónico, como un teléfono móvil o un ordenador. Esto es debido al auge de la digitalización en la actualidad en nuestro día a día ya que, por ejemplo, es mucho más fácil buscar información sobre un tema mediante un dispositivo electrónico que buscarla en un libro. Puede que la información obtenida no sea tan valiosa, pero si más rápida.

Este hecho se ha expandido en muchos sectores. Por ejemplo, un gran avance ha sido la codificación de barras para los productos que compramos. Las grandes tiendas poseen una gran cantidad de artículos, y por medio de esta codificación podemos saber cuantas unidades se han vendido, así como proporcionarnos rápidamente el precio del artículo y hacer que la compra sea más rápida.

Todo esto nos aporta información valiosa, la cual debe procesarse adecuadamente. A partir de esto, surge la minería de datos. La minería de datos consiste en identificar información útil en los datos, enfocándose en parámetros o patrones que siguen. Para ello, se usa tanto la inteligencia artificial como bases de datos y métodos estadísticos. Conocer estos parámetros o patrones puede ser de gran utilidad para las grandes empresas, ya que a partir de ellos pueden obtener un gran beneficio o cumplir sus objetivos si saben como usarlos de manera correcta.

La minería de datos no solo se utiliza en negocios, en medicina también es importante. Puede contabilizar los diagnósticos que se les ha dado a los pacientes para en un futuro saber cómo proceder si hay un paciente con síntomas parecidos. Del mismo modo que se usa para la medicina, se aplica en la mejora de la seguridad de los alimentos o de detectar delincuentes.

Para un buen análisis de la minería de datos se deben seguir una serie de pasos y así abarcar con todo lo necesario para cumplir los objetivos. Existen varias metodologías, pero la más común se denomina CRISP-DM, la cual es utilizada para este proyecto.

Las siglas de CRISP-DM corresponden a *Cross-Industry Standard Process for Data Mining*. Se basa en un modelo que consta de seis fases previstas como un proceso cíclico, y puede variar el orden de las fases dependiendo del proyecto a realizar.

Las seis fases mencionadas anteriormente son las siguientes:

Comprensión de problema. Esta fase es la base del proyecto, en la que se determinan los objetivos, se evalúa la situación actual y cómo se desarrolla el plan a proceder. Los objetivos en términos comerciales se basan en los tipos de clientes a los que van dirigidos los productos.

Comprensión de los datos. En esta fase, una vez determinados tanto los objetivos como el desarrollo del plan a proceder, se hace la recopilación de datos en la que se añade una descripción de ellos y posteriormente se verifican la calidad de los datos.

Preparación de datos. Una vez realizada la primera verificación se procede a limpiar, seleccionar y transformar a la forma deseada los datos para ajustarlos a las técnicas que queremos utilizar.

Modelaje. En esta etapa se analizan y estudian los datos para crear los modelos de predicción más idóneos acorde con los objetivos preestablecidos. Al escoger el modelo a utilizar se debe definir el método, para la posterior validación del modelo y verificación de que los resultados sean sólidos. La validación se realiza utilizando datos que no se han usado para la construcción del modelo.

Evaluación. Al obtener los resultados de los modelos escogidos, se observa su fiabilidad mediante técnicas de evaluación. Los resultados deben analizarse en base a los objetivos propuestos en la primera fase. Se pueden dar casos en los que se debe volver a fases anteriores para una mejor predicción, cosa que suele suceder cuando se descubren nuevos patrones en los datos.

Implementación. La información obtenida a partir de los modelos creados se debe monitorizar para detectar posibles cambios, ya que existen casos en los que el cliente puede cambiar de opinión. A partir de la información se implementa el modelo para alcanzar el objetivo del proyecto.

2.1. Objetivos del proyecto

El objetivo principal del presente trabajo es predecir la popularidad de noticias en internet, para ello se utilizará la minería de datos. Se pretende crear modelos para usarlos con el fin de determinar así la popularidad, enfocándose en los modelos *Random Forest* y *Regresión Logística*. Otro de los objetivos es investigar estos modelos de predicción, para saber cómo actúan cada uno de ellos en función de los datos, y en especial ver cómo el *Random Forest* varía su tasa de acierto al cambiar los valores de sus parámetros. Además, al realizar dos modelos se compararán entre sí para observar su tasa de acierto y por qué uno estima mejor que otro.

Estos datos poseen columnas las cuales aportan diversas características que describen diferentes aspectos como palabras clave, contenido digital o referencias a otros artículos. Se pretende dividir la columna “shares” en categorías, y el proyecto se centrará en la categoría de alta popularidad. El hecho de saber qué publicaciones son las que tendrán una mayor popularidad es muy beneficioso para las empresas ya que a partir de estas podrán obtener un número mayor de seguidores y visitantes, lo que conlleva a la posibilidad de lograr nuevos patrocinadores y así más recursos económicos. Las empresas quieren que los trabajadores aprovechen al máximo el tiempo de trabajo, así obtener una gran productividad y lograr los objetivos propuestos. No obstante, podría haber otros tipos de perspectivas respecto a las categorías en las cuales centrarse.

Por otra parte, se pretende aplicar la metodología CRISP-DM para aprender cómo se utiliza de forma práctica y poder desarrollar bien el estudio teniendo todos los factores en cuenta. El último objetivo es adquirir conocimientos de programación con Pandas y *Sklearn* mediante la plataforma Anaconda en lenguaje Python. Para llevar a cabo este proyecto se aplicarán estas librerías, ya que proporcionan herramientas para poder manipular los datos de forma rápida y, también crear modelos de predicción de manera sencilla, los cuales se puedan entender sin dificultad.

2.2. Abasto del proyecto

Tal como se menciona en los objetivos, se desea aplicar CRISP-DM en este estudio. Por otro lado, al no poseer del todo el tiempo deseado el proyecto no se centrará en todas las fases de esta metodología, sino en las siguientes:

Comprensión del problema. Esta fase es fundamental para este proyecto ya que se podría enfocar de varias formas. En este caso, al haber realizado practicas en diferentes empresas se entiende cuáles son los objetivos de estas y las dificultades que conlleva conseguirlos. De este modo se considera que se dispone del conocimiento suficiente para realizar este proyecto.

Comprensión y preparación de los datos. Al haber realizado la toma de datos una empresa externa, es fundamental entender a qué se refiere cada columna y cómo interpretarla. Se construirán dos tipos de datos; unos donde se separarán los *shares* en tres categorías y otro en dos categorías. Se crearán estos dos tipos de datos para saber cómo pueden influir en que el modelo sea más preciso o menos. Una vez se sepa el nivel de influencia de cada variable en el modelo *Random Forest*, se añadirán columnas nuevas siendo combinaciones de otras. Todo se programará el lenguaje Python para hacerlo más simple y rápido.

Modelaje y evaluación. Una vez creados los datos, se procederá a estudiarlos mediante modelos de predicción, específicamente con los modelos *Random Forest* y *Regresión Logística*. Una vez creado el modelo *Random Forest* se irán cambiando sus parámetros, y posteriormente se validará con datos que no se han utilizado para crear el modelo, para ver cómo va variando su tasa de acierto respecto a noticias con alta popularidad. Luego se compararán los resultados obtenidos de los dos modelos de predicción.

Faltaría la última fase que es la implementación, pero no se puede llevar acabo ya que no se dispone del tiempo suficiente. Implicaría poner a prueba los modelos predictivos durante un año para ver si los resultados son significativos.

2.3. Herramientas utilizadas

A continuación, se detallan las herramientas que se han utilizado para la realización de este proyecto.

2.3.1. Python

Python es un lenguaje de alto nivel preparado para diferentes finalidades, tanto para aplicaciones Windows a servidores de red o páginas web. Es un gran lenguaje para comenzar a programar ya que posee un código más legible, además de que se puede obtener gratuitamente. Por otra parte, no son necesarias tantas líneas de programación a diferencia de otros lenguajes, ya que Python puede llegar a tener 3 o 5 líneas de código menos que su equivalente en Java o C. Cabe decir que permite realizar muchas tareas sin necesidad de tener que programarlas des de cero.

Python contiene múltiples librerías para desarrollar programas con un gran número de funciones y diferentes temáticas. Estas librerías son paquetes que contiene funciones para facilitar a la hora de crear nuevos sistemas. Se puede desarrollar tanto en Windows o Unix como en OS u otros.

Aporta una gran cantidad de información en caso de duda mediante páginas web, donde se aclaran todas sus funciones aportando ejemplos. A parte de esto, existen foros donde se pueden corregir errores comunes a la hora de programar, y se han redactado artículos sobre este lenguaje, los cuales ejemplifican todo el potencial que contiene.

Además de ser uno de los más utilizados y disponer de librerías para crear modelos predictivos de minería de datos, es de gran ayuda haberlo estudiado durante el grado ya que no hace falta aprender a programarlo des de cero y así lograr un gran rendimiento del lenguaje.

2.3.2. Anaconda y Spyder

Anaconda es una distribución libre y abierta tanto del lenguaje Python como el lenguaje R. Esta distribución se utiliza en ciencia de datos y, como en este proyecto, en aprendizaje automático. Es muy utilizada alrededor del mundo, llegando a un número aproximado de 6 millones de usuarios, ya que al ser una multiplataforma se ha instalado en sistemas Linux, MacOs y Windows.

Anaconda permite procesar grandes volúmenes de datos a gran velocidad, llegando a realizar a la vez análisis predictivos y cálculos científicos. Además, proporciona con facilidad la instalación y administraciones de entornos y paquetes para el estudio de datos, brindando a los usuarios recursos de aprendizaje avanzados. La distribución descrita aporta un despliegue de softwares donde se encuentran editores.

Spyder es un entorno de desarrollo integrado gratuito (IDE) el cual se incluye en Anaconda. Esta diseñado para científicos, ingenieros y analistas de datos. Proporciona funciones avanzadas de edición, prueba interactiva y depuración de variables. Las siglas significan *Scientific Python Development Environment*.

Spyder ofrece una cantidad muy extensa de paquetes científicos populares, en los cuales se destacan *NumPy*, Pandas y *Sklearn*. Concede un *editor avanzado* donde se pueden crear varias pestañas a la vez e ir programando. También aporta una *consola interactiva*, donde se pueden ejecutar todas las funciones creadas, ver los resultados o parar de ejecutar una función; y una *pestaña* donde se almacenan las variables creadas, las cuales puedes ver, editar o sencillamente eliminarlas sin tener que ejecutarlas en la consola. Además, proporciona un *historial* para observar las comandas ejecutadas.

2.3.3. Pandas

Pandas es una librería de Python la cual permite la manipulación y analizar datos. Proporciona estructuras de datos para poder seleccionar, manipular o transformar tablas numéricas.

Los datos se organizan en *Series* que se unen creando *DataFrames*, los cuales se utilizan para la manipulación con indexación integrada. Puede leer y escribir datos en varios tipos de formatos de archivos, un ejemplo es Excel.

Posibilita reestructurar datos y la segmentación del conjunto de ellos tanto vertical como en horizontal. Se puede insertar nuevas columnas o eliminarlas. Permite realizar una gran serie de operaciones sobre las columnas, tanto sumar como dividir. Esto facilita combinar los datos y poder crear nuevas columnas a partir de estas combinaciones.

Es una herramienta poderosa con muchos usos, la cual es bastante eficaz para el tratamiento de datos. Al ser bastante sencilla, su uso es óptimo para estudiantes que están empezando a programar en el ámbito de la minería de datos.

2.3.4. Sklearn

Scikit-learn, también llamado *Sklearn*, es una biblioteca de aprendizaje automático para lenguaje Python. Esta biblioteca esta incluida en la plataforma Anaconda, y cuenta con algoritmos tanto de clasificación como de regresión y agrupación. Está diseñado para operar con las bibliotecas numéricas y científicas de Python.

Facilita mucho el trabajo ya que aporta funciones que simplifican el código de programación y puedes enfocar el trabajo en analizar mejor los resultados obtenidos. Los algoritmos de clasificación son muy útiles de manera que se puede observar rápidamente cómo evoluciona el modelo creado, viendo dónde tiene más fallos y más aciertos. Proporciona una gran cantidad de parámetros importantes para tener todos los factores en cuenta, como por ejemplo la *Precision*.

3. Comprensión y preparación de los datos

La comprensión de datos es una fase muy importante en este proyecto. Al haber realizado la recopilación de datos una empresa externa, se tiene que tener claro qué significan todas las columnas que han creado y han creído que eran parámetros que podrían ser significativos para la predicción de la popularidad de las noticias.

Los datos fueron proporcionados por la página web llamada *Mashable* en la base de datos pública *UC Irvine*, se puede acceder a través del link adjuntado en la bibliografía. Los datos poseen 60 columnas y 39640 filas y se obtuvieron en formato csv.

Mashable es un blog digital donde se publican noticias, fundado por Pete Cahsmore, siendo uno de los blogs más populares en toda la red. Las noticias que publica abarcan diferentes temas, por ejemplo, relacionados con redes sociales o empresas tecnológicas como Apple. Esta página desea saber antes de publicar una noticia en su blog si será popular para así conseguir un número mayor de lectores y continuar siendo una de las más influyentes.

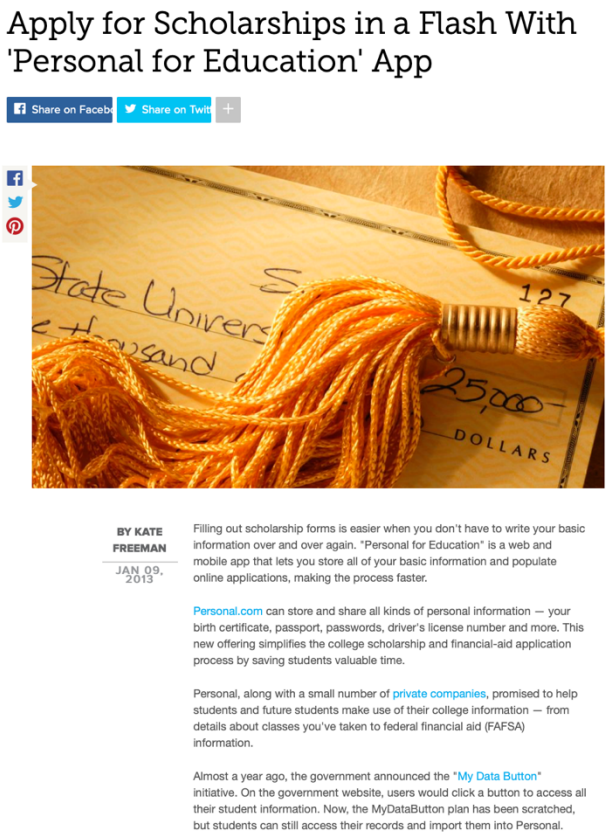

Figura 1. Ejemplo de noticia publicada en Mashable

3.1. Datos iniciales

A continuación, se explica el significado que tiene cada una de las columnas:

-. Url: Indica la Url de la noticia

-. Timedelta: Días entre el día de la publicación de la noticia y la toma de datos.

0. N_tokens_title: Número de palabras en el título

1. N_tokens_content: Número de palabras en la noticia

2. N_unique_tokens: Proporción de palabras únicas en la noticia

3. N_non_stop_words: Proporción de palabras no vacías en la noticia

4. N_non_stop_unique_tokens: Proporción de palabras no vacías únicas en la noticia

5. Num_hrefs: Número de links en la noticia

6. Num_self_hrefs: Número de links de otras noticias de la misma web

7. Num_imgs: Número de imágenes en la noticia

8. Num_videos: Número de vídeos en la noticia

9. Average_token_length: Promedio de la longitud de las palabras en la noticia

10. Num_keywords: Número de palabras clave en la noticia

11. Data_channel_is_lifestyle: Indica si la noticia trata sobre el estilo de vida personal

12. Data_channel_is_entertainment: Indica si la noticia trata sobre ocio

13. Data_channel_is_bus: Indica si la noticia trata sobre negocios

14. Data_channel_is_socmed: Indica si la noticia trata sobre redes sociales

15. Data_channel_is_tech: Indica si la noticia trata sobre las nuevas tecnologías

16. Data_channel_is_world: Indica si la noticia trata sobre el planeta

17. Kw_min_min: Indica el mínimo número de comparticiones entre todas las noticas en las que ha escrito la peor palabra clave de la noticia actual

18. Kw_max_min: Indica el máximo número de comparticiones entre todas las noticas en las que se ha escrito la peor palabra clave de la noticia actual

19. Kw_avg_min: Indica el promedio entre el máximo y el mínimo todas las noticias en las que se ha escrito la peor palabra clave de la noticia actual

20. Kw_min_max: Indica el mínimo número de comparticiones entre todas las noticas en las que ha escrito la mejor palabra clave de la noticia actual

21. Kw_max_max: Indica el máximo número de comparticiones entre todas las noticas en las que ha escrito la mejor palabra clave de la noticia actual

22. Kw_avg_max: Indica el promedio entre el máximo y el mínimo todas las noticias en las que se ha escrito la mejor palabra clave de la noticia actual

23. Kw_min_avg: Indica el promedio del número de comparticiones entre todas las noticias en las que se ha escrito la peor palabra clave de la noticia actual

24. Kw_max_avg: Indica el promedio del número de comparticiones entre todas las noticias en las que se ha escrito la mejor palabra clave de la noticia actual

25. Kw_avg_avg: Indica el promedio del número de comparticiones entre todas las noticias en las que se ha escrito alguna de las palabras clave de la noticia actual

26. Self_reference_min_shares: Indica el número mínimo de comparticiones de la noticia menos popular de las que hace referencia la noticia actual

27. Self_reference_max_shares: Indica el número máximo de comparticiones de la noticia más popular de las que hace referencia la noticia actual

28. Self_reference_avg_sharess: Indica el promedio de comparticiones de la noticia de las que hace referencia la noticia actual

29. Weekday_is_monday: Indica si la noticia ha sido publicada un lunes

30. Weekday_is_tuesday: Indica si la noticia ha sido publicada un martes

31. Weekday_is_wednesday: Indica si la noticia ha sido publicada un miércoles

32. Weekday_is_thursday: Indica si la noticia ha sido publicada un jueves

33. Weekday_is_friday: Indica si la noticia ha sido publicada un viernes

34. Weekday_is_saturday: Indica si la noticia ha sido publicada un sábado

35. Weekday_is_sunday: Indica si la noticia ha sido publicada un domingo

36. Is_weekend: Indica si la noticia ha sido publicada en fin de semana

37. LDA_00: Indica la cercanía de la noticia al tema principal 0

38. LDA_01: Indica la cercanía de la noticia al tema principal 1

39. LDA_02: Indica la cercanía de la noticia al tema 2

40. LDA_03: Indica la cercanía de la noticia al tema 3

41. LDA_04: Indica la cercanía de la noticia al tema 4

42. Global_subjectivity: Indica la subjetividad de la noticia

43. Global_sentiment_polarity: Indica la polaridad de la noticia

44. Global_rate_positive_words: Ratio de palabras positivas en la noticia

45. Global_rate_negative_words: Ratio de palabras negativas en la noticia

46. Rate_positive_words: Ratio entre palabras positivas y palabras no neutrales

47. Rate_negative_words: Ratio entre palabras negativas y palabras no neutrales

48. Avg_positive_polarity: Promedio de polaridad en palabras positivas

49. Min_positive_polarity: Mínima polaridad en palabras positivas

50. Max_positive_polarity: Máxima polaridad en palabras positivas

51. Avg_negative_polarity: Promedio de la polaridad en palabras negativas

52. Min_negative_polarity: Mínima polaridad en palabras negativas

53. Max_negative_polarity: Máxima polaridad en palabras negativas

54. Title_subjectivity: Indica la subjetividad del titulo

55. Title_sentiment_polarity: Indica la polaridad del titulo

56. Abs_title_subjectivity: Indica la subjetividad de la noticia en valor absoluto

57. Abs_title_sentiment_polarity: Indica la polaridad de la noticia en valor absoluto

-. Shares: Número de comparticiones de la noticia

3.2. Preparación de los datos

3.2.1. Limpieza de datos

En esta fase se eliminaron las columnas las cuales no servían para predecir ya que eran variables no predictivas. Se decidió eliminar las dos primeras columnas, timedelta y la url.

La url no tenía valor ya que solo se utilizaba para ver la noticia y no saber el nivel de popularidad que podría tener.

Timedelta no era útil, no se tenían en cuenta cuanto tiempo había transcurrido desde la publicación de la noticia hasta la toma de datos. Se podría dar el caso en el que aumentando el tiempo la noticia podría haber obtenido más comparticiones.

3.2.2. Exploración de *missing values*

Los *missing values* son posiciones donde debería de haber valores, pero no aportan ninguna información. Estos valores se llaman así porque no existen o no se han registrado cuando se ha realizado la recopilación de datos. Es importante identificar los *missing values* ya que *sklearn* no los acepta como datos cuando se crean los modelos de predicción, en el caso que se utilice un *missing value* aparece un error y no se crea ningún modelo. Por esta razón es fundamental comprobar si en los datos que se usaran hay *missing values*.

Una vez leídos los datos en formato csv u otro, se introducen en un *Dataframe* para poder utilizar la librería Panda, si hubiese un valor vacío toman un valor de *NaN* en el *Dataframe*.

Para saber si en los datos había *missing values* hay una función de la librería de pandas, la cual te informa una por una si en esa columna hay *missing values*. Se utilizó esta función con los datos proporcionados y no se obtuvo ningún *missing values*.

3.3. Transformación de datos

3.3.1. Dividir columna *shares* por categorías

Como se ha comentado en el apartado de objetivos, se desea predecir las publicaciones que obtendrán un número alto de *shares*, publicaciones con una popularidad alta. En esta fase del proyecto se tuvo que escoger formas de dividir la columna “shares” en categorías para utilizarla como columna *label* o respuesta.

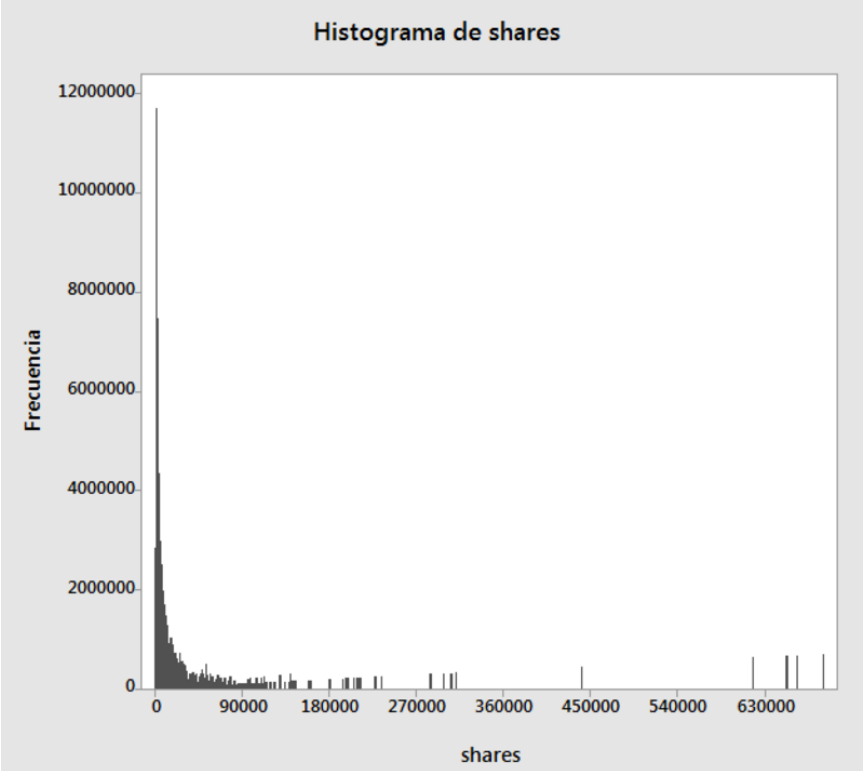

Figura 2. Histograma de la columna shares de los datos

Como se puede observar hay una gran cantidad de datos con los cuales lograron un pequeño número de comparticiones.

Se decidió realizar dos tipos de datos, los cuales se diferenciaban al tener uno, dos categorías para saber la popularidad de las noticias y el otro tipo, tres categorías. El objetivo era tener la misma cantidad de datos para cada una de las categorías en el primer tipo de datos.

En los datos con dos categorías, se obtuvo una buena división en el valor de comparticiones de 1400. De 0 comparticiones a comparticiones inferiores a 1400 se consideraban como popularidad baja y igual a 1400 o superior se consideraban como noticias con una popularidad alta.

En cambio, con los datos de tres categorías, al tener que realizar más divisiones, se tuvieron que hacer más experimentos durante más tiempo ya que era más costoso ver bien la separación. Se decidió hacer las divisiones de tal manera que la categoría de media popularidad tuviese el doble de datos que la categoría de baja y alta. De este modo se procedió a hacer la división en 950 y 2700 comparticiones. Noticias con comparticiones inferiores o igual que 950, se consideraban como popularidad baja. Noticias con comparticiones superiores a 950 pero noticias con un número inferior o igual de 2700 se consideraban como popularidad media. Por último, las noticias con un número de comparticiones superior de 2700 se consideraban como popularidad alta.

4. Modelaje y Evaluación

Al obtener los datos bien estructurados y con el formato que se deseaba se procedió a escoger el modelo predictor y proceder a su validación.

4.1. Modelos escogidos

Un modelo, en minería de datos, se entiende como una representación que intenta capturar los patrones y relaciones entre los datos, a partir de la información obtenida realiza predicciones para diferentes casos. En términos estadísticos, se entiende como una ecuación, como en *Regresión Logística*, que pretende reproducir los casos observados de la forma más parecida posible. A partir de estos casos se modela la ecuación, es decir, se determinan los diferentes coeficientes que debe tener cada una de las variables.

A continuación, se explicarán los dos modelos usados para obtener un mejor conocimiento y saber como actúa cada uno de ellos. Además, se detallará en el caso de *Random Forest* los parámetros que se han considerado relevantes para modificar y no dejarlos por defecto.

Estos dos modelos usan el aprendizaje supervisado, se denomina así ya que los datos para el entrenamiento del modelo incorporan la solución deseada, la cual se llama *label* o respuesta.

4.1.1. *Regresión Logística*

La *Regresión Logística* es un algoritmo supervisado utilizado tanto en el sector de ciencias medicas y sociales, es una técnica la cual predice variables categóricas. Se puede ampliar su conocimiento, por ejemplo, en como determinar si una imagen puede contener un tipo de animal específico. Si se desea el modelo permite obtener las variables que influyen del modelo. Estas variables pueden ser binarias o continuas.

Este tipo de regresión se compone de dos partes, la primera es la suma de todas las variables [x] de los datos multiplicadas cada una por un coeficiente [w]. La segunda se basa en aplicar la función logística al resultado obtenido. La expresión matemática sería la siguiente:

$$y = \sigma(z) = \sigma(w_0 x_0 + w_1 x_1 + \cdots + w_n x_n)$$

La función logística es la que a continuación se muestra:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Esta función está acotada entre 0 y 1, estando 0 y 1 incluidos. Sus resultados se interpretan como probabilidades. En el caso de tener dos categorías y obtener un valor de 0.85, significa que hay una probabilidad del 0.85 de que sea grupo 1. Los valores obtenidos menores de 0.5 correspondían al grupo 0 y valores superiores a 0.5 correspondían al grupo 1.

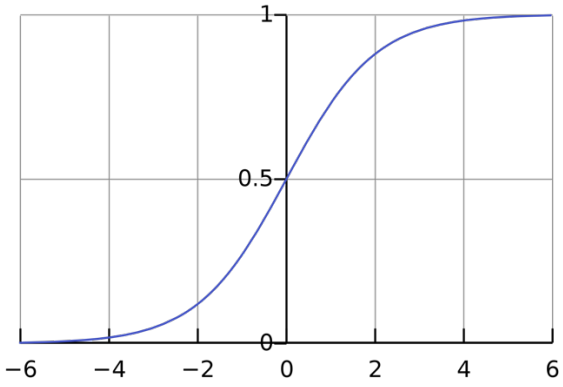

Figura 3. Gráfico donde se representa la función logística

En el caso de tener más de dos categorías se utilizaría la regresión logística multinominal. Combina diferentes funciones logísticas a la vez.

4.1.2. Árbol de decisión

A continuación, se detallará el funcionamiento de un árbol de decisión para así entender mejor como funciona *Random Forest*.

Los árboles de decisión son un modelo de aprendizaje supervisado, igual que *Regresión Logística,* el cual se usa para problemas de clasificación. Las variables de entrada pueden

ser tanto categóricas como continuas. Se dedica a tomar una serie de decisiones en forma de árbol. Esto lleva a tener dos partes a la hora de representarlo.

Está el nodo, siempre es creado a partir de una variable de los datos de entrenamiento, va acompañado de una condición. Hay dos tipos de nodos:

- Nodo de decisión: poseen una condición y tienen dos ramas que salen de él. Se interpreta como una pregunta que se le hace al ejemplo que el modelo está analizando.
- Nodo de predicción: este nodo determina la decisión que el modelo decide para el ejemplo que haya analizando. No posee ningún nodo por debajo, es decir, no sale ninguna rama de él y además no posee ninguna condición.

Por otro lado, están las ramas, que son elementos que sirven para unir los nodos. De cada nodo salen dos, según si los datos que pasan por ese nodo cumplen la condición o no la cumplen.

Los nodos proporcionan la información siguiente:

- Condición: el modelo predictor crea una condición para cada variable tal y como se ha comentado.
- Entropía de Shannon: proporciona valores entre 0 y 1. Un valor igual a 1 significa que los datos a explorar están bien distribuidos, es decir, los datos poseen el mismo número de ejemplos de cada posible clasificación. Por otro lado, un valor igual a 0 significa que los datos a explorar son completamente homogéneos, pertenecen a un único tipo de clasificación.
- Samples: indica el número de muestras que cumplen con la condición para llegar al nodo.
- Value: número de muestras de cada clase que llega a ese nodo.
- Class: la categoría que se le determina al ejemplo una vez llegado a ese nodo.

El modelo escoge el primer nodo a partir de la condición que tenga la entropía más alta. De este modo los nodos que van por debajo de este, tienen valores de entropía inferiores.

Siendo el último nodo del árbol el que tenga una entropía de valor 0. Esto se puede modificar a partir de parámetros que el modelo posee, como limitar la profundidad del árbol.

Mediante la librería *sklearn* se puede modificar los árboles de decisión para que sus parámetros no tengan el valor por defecto y no estén regularizados. Regularizar los parámetros significa limitar las capacidades del modelo para obtener un nuevo modelo de predicción más simple y que generalice mejor.

Por la forma en que se crea el árbol provoca que sea inestable, primeramente el árbol escoge una variable de las proporcionadas para crear una condición. El árbol elije la mejor condición en ese momento, pero a la larga puede que no acabe siendo la correcta. En definitiva, es un tipo de algoritmo inestable ya que cambiando la variable con la que se comienza o variando los datos de entrenamiento, puede variar bastante el árbol resultante.

Si no se limitan se pueden dar casos en el que en modelo sea muy complejo y se ajuste bien a los datos de entrenamiento, pero mal a los datos de prueba.

A continuación, se explicarán los parámetros que se han modificado en este proyecto para los árboles de decisión:

- **Max_depth**: proporciona la profundidad máxima del árbol a crear. El valor por defecto de este parámetro es que el árbol se desarrolle tanto como le sea posible. La profundidad indica el número de nodos que hay hasta llegar a un nodo de predicción.
- **Min_samples_leaf**: indica el número mínimo de muestras que debe haber en un nodo para poder ser utilizado en el modelo. Por defecto el número mínimo es de 2.

4.1.3. Random Forest

Random Forest forma parte de la familia de los métodos de *ensemble* o de combinación. Los métodos *ensemble* utilizan múltiples algoritmos de aprendizaje para lograr predicciones que mejoran las que se obtendrían con solo un algoritmo de aprendizaje. Estos métodos se pueden aplicar en diferentes tipos de modelos de aprendizaje, no se limitan solo a árboles de decisión.

Random Forest es una técnica muy popular ya que se ha utilizado en una gran cantidad de proyectos. Es una combinación de árboles de decisión en los cuales no se utilizan todas las variables del modelo ni todas las filas de datos.

Consiste en hacer que los árboles de decisión se creen a partir de subconjuntos de los datos originales divididos aleatoriamente usando solo un número limitado de variables y filas, es decir, entre todos los árboles de decisión que puede crear *Random Forest* ninguno ve todos los datos destinados para entrenar el modelo, aunque se pueden dar casos en los que algún dato este incluido en varios subconjuntos a la vez. La ventaja que aporta este método es que los árboles se entrenan con diferentes partes de los datos de entrenamiento para los mismos datos de prueba, logrando árboles que tienen diferentes puntos de vista.

Una vez creados todos los árboles, el algoritmo junta los resultados obtenidos, haciendo que errores que hayan podido tener algunos modelos se corrijan automáticamente, logrando un modelo con un nivel de generalización muy alto.

Para combinar los resultados obtenidos de cada árbol usa una técnica llamada *Soft-voting* (voto suave). *Soft-voting* es una técnica la cual se da más valor a los resultados procedentes de árboles con mejores métricas de evaluación.

Además de modificar varios parámetros de los árboles de decisión de dentro de *Random Forest*, también se han modificado parámetros de *Random Forest*, para tener un mejor modelo. Los parámetros son los siguientes:

- **N_estimators**: este parámetro sirve para introducir el número de árboles de decisión que va a tener el modelo de *Random Forest* a crear. Un número alto es mejor ya que va mejorando el modelo, pero al llegar aun cierto número el modelo se ralentiza. El valor por defecto es de 100 árboles.

- **Max_features**: sirve para que los árboles de decisión se entrenen con muestras aleatorias diferentes de los datos. Indica el número de variables que se deben de considerar en cada árbol. Por defecto el modelo hace la raíz cuadrada de el número total de variables que tiene los datos y este lo utiliza como el número de variables a considerar.

- **Random_state**: sirve para que a la hora de hacer diferentes experimentos la aleatorización para escoger los datos sea la misma.

4.2. Validación del modelo

La validación es un proceso muy importante para la minería de datos, se entiende como un proceso donde se analiza el rendimiento del modelo y se observa si es aceptable, es decir, permite predecir cómo funcionará el modelo con conjuntos de datos no utilizados para crearlo. Hay diferentes tipos de validación, que se detallan seguidamente pero antes se explicará *Overfitting*.

4.2.1. Overfitting

Overfitting, el cual significa sobreajuste, es un factor que hay que tomar en consideración ya que puede tener una gran influencia en los modelos. Se entiende como el efecto de un exceso de entreno del modelo de predicción.

El modelo debe tener la capacidad de poder predecir el resultado de otros casos a partir de los datos con los que ha sido entrenado, de tal manera que pueda solucionar situaciones diferentes a las presentes durante el periodo de entreno. Cuando el modelo se sobreentrena o se entrena con una cantidad de datos poco comunes, el algoritmo puede ajustarse a unas características muy específicas, cosa que tenían los datos con los que se ha entrenado. El modelo intenta ajustarse demasiado a estos datos inexactos, provocando una reducción en el acierto de sus predicciones.

Esto se puede observar cuando se evalúa el modelo con los datos de entreno y los de prueba, si el modelo predice mucho mejor los datos de entreno que los datos de prueba nuestro modelo está sobre ajustado. El modelo debe de ser capaz de generalizar.

Overfitting no depende solo de los parámetros y datos, sino también de la estructura del modelo con la forma de los datos. Para reducir la posibilidad de que aparezcan estos dos factores, se limita el modelo para que no sea muy complejo.

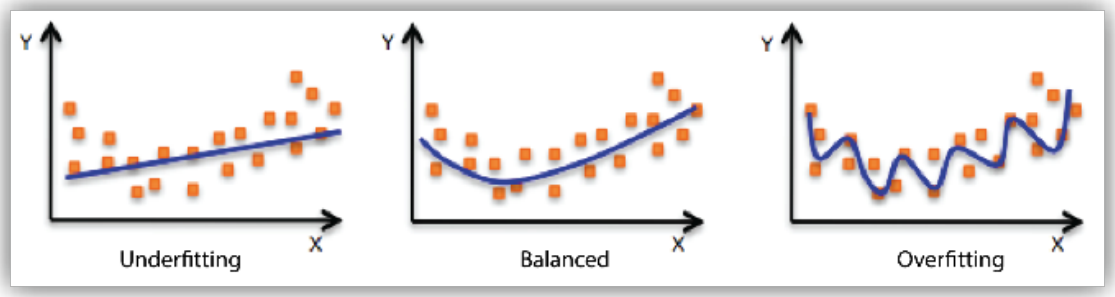

Figura 4. Representación de Overfitting

4.2.2. *Holdout validation*

En este método se procede a dividir el grupo de datos en dos tipos: datos de prueba y datos de entrenamiento.

Los datos de entrenamiento son los que se usan para modelar los modelos predictores que se escogerán, es decir, los datos a partir de los cuales los algoritmos de predicción aprenden. Esto es muy importante ya que es la base de los modelos, a partir de estos datos los modelos se ajustan y deben ser capaces de saber hacer nuevas predicciones a partir de otros datos.

Los datos de prueba son los que se usan para ver como los modelos predicen, es decir, sirven para saber cuál es el porcentaje de acierto de los modelos. Estos tipos de datos son los que nunca los modelos habrán visto, entonces son bastante fiables las predicciones que hagan al ser nuevos casos y es a partir de donde se obtienen las métricas de evaluación.

Se dividen los datos para así no utilizar en la etapa de validación los que se ha usado para la creación de los modelos. Si se utilizaran los mismos datos, el modelo ya sabría la respuesta a dar y obtendríamos buenos resultados, pero estaríamos haciendo una validación errónea.

Este tipo de validación se emplea mediante una función, llamada *train_test_split,* la cual permite seleccionar el porcentaje de datos que se desea tener como datos de prueba y datos de entrenamiento. Esta selección se hace aleatoriamente y además permite hacer al mismo tiempo una buena distribución de la cantidad de cada categoría en los dos tipos de datos.

4.2.3. Cross-validation

La *cross-validation* o validación cruzada es un método de validación que se encarga de dividir los datos de forma que se obtenga las mejores predicciones de los modelos predictivos escogidos y así minimizar el error en la fase de validación del proyecto. Es de gran ayuda ya que asegura estar ejecutando de forma adecuada el modelo. Hay diversos tipos de validación basadas en esta técnica:

- ***K-fold validation o validación cruzada de k iteraciones***. Este tipo de validación cruzada es un método muy popular. Mezcla los datos y los divide en k número de grupos. Una vez creados los grupos escoge uno de ellos y lo utiliza como datos de prueba, los otros restantes se utilizan como datos de entrenamiento del modelo, para poder ajustar y posteriormente evaluar el modelo. Esto se repite para cada grupo, siendo cada grupo una vez de las k veces los datos de prueba. Acaba haciendo un número de k modelos. Para acabar el método realiza una media de las métricas de evaluación obtenidas de cada modelo construido.

- ***Leave One Out Cross Validation (LOOCV).*** LOOCV significa validación cruzada dejando uno fuera. Divide los datos de manera que en cada iteración cree un modelo donde los datos de prueba solo sean una muestra y la cantidad restante de datos la toma como datos de entrenamiento. En este tipo de validación el conste computacional es mayor ya que es necesario un gran número de iteraciones para crear bastantes modelos. Una vez creados los modelos procede igual que el tipo de validación descrito anteriormente.

- ***Leave-P-Out Cross Validation.*** Este tipo de método para cada modelo deja p cantidad de datos como datos de prueba y los restantes como datos de entrenamiento del modelo. Esto se repite tantas veces como la muestra original se

pueda separar de esta forma. Posteriormente se promedia las medidas de evaluación de todos los modelos creados.

- ***Stratified K-forld cross validation***. Se pueden dar situaciones en las que los datos no estén bien distribuidos. Un ejemplo sería tener en los datos de prueba una gran cantidad de publicaciones con una popularidad alta y en los datos de entrenamiento muy pocos. El modelo al no haber tenido suficientes datos con popularidad alta no dará predicciones correctas para este tipo. Para solucionar este tipo de problemas, este método de validación hace la división de manera que el conjunto de pruebas y entrenamiento contengan el mismo porcentaje de cada clase de los datos totales. Una vez creados los modelos hace la media de las métricas de evaluación obtenidos de cada uno.

- ***Aleatori cross validation.*** Este método se diferencia de los mencionados anteriormente, al dividir aleatoriamente el conjunto de datos en datos de entrenamiento y datos de prueba. Esto sucede de manera iterativa hasta el número que se desee, y para cada iteración se crea un modelo nuevo. Al final se hace la media de todas las métricas de evaluación de cada modelo. El inconveniente de este tipo de validación cruzada, es que se pueden dar casos donde no se lleguen a utilizar datos para entrenamiento o se repitan datos de entrenamiento.

La validación cruzada también se puede utilizar para poder obtener los parámetros de los modelos que proporcionan mejores predicciones. Se procedería usando primeramente el método *holdout* para así tener los datos de entrenamiento y los datos de prueba. Después se realizaría la validación cruzada sobre los datos de entrenamiento. En esta fase se conseguirían los parámetros y luego se pondrían en práctica a partir de los datos de prueba.

4.2.4. GridSearch

GridSearch es una función de la librería *Sklearn*, la cual se utiliza para obtener los mejores parámetros de diversos tipos de modelos a la vez. Una vez realizado proporciona los resultados obtenidos.

Primero se introduce a la función los tipos de modelos que son con los que se desea realizar predicciones y parámetros con los diferentes valores que se quieren probar de cada

modelo. Después *GridSearch* realiza experimentos para informar sobre los mejores valores para los parámetros, entre los que se ha introducido de cada modelo, y así obtener sus métricas de evaluación. Una vez se consiguen los mejores, compara los modelos para saber cuál es el mejor y se ajusta más a los datos proporcionados. Usa *crossvalidation* para cada uno de los modelos y parámetros, obteniendo una gran eficacia en sus resultados.

4.2.5. Validación escogida

GridSearch se intento poner en práctica en este proyecto, pero después de unos días de espera se decidió no utilizarla. No se obtuvo ningún resultado al necesitar más tiempo del cual no se disponía. No obstante, es una función muy útil si se dispone de más potencia computacional al facilitar mucho el trabajo de programación.

Le valoró usar Validación cruzada en este proyecto, pero al no disponer de toda la potencia necesaria y el tiempo suficiente, se decidió no utilizarla. Hace falta una gran cantidad de tiempo, no tanto en comparación con *GridSearch*, para conseguir los resultados.

De este modo se escogió el método *Holdout* validation. Este método se ha usado para el proyecto, distribuyendo un 30% de los datos totales para datos de prueba y el 70% para datos de entrenamiento. Una función que tiene *train_test_split* es que permite distribuir de manera que tanto los datos de prueba como los datos de entrenamiento contengan porcentajes parecidos de las categorías del conjunto total de datos, la cual se ha utilizado en el proyecto para así conseguir un modelo más óptimo.

4.3. Métricas de evaluación

A continuación, se explicará las métricas utilizadas para evaluar los resultados obtenidos de los modelos predictivos escogidos. Se pueden tener diversos enfoques, pero tal y como se explica en el apartado de objetivos, este proyecto se centra en obtener el mayor porcentaje de acierto al predecir las noticias con una popularidad categorizada como alta. Para poder tener en cuenta varios factores a la vez se focalizará en *F1-score*. Tanto este factor como la *Accuracy score*, *Precision*, *Recall* y *Confusion Matrix*, se detallarán para poder tener un mayor conocimiento de ellos. Estos factores se consiguen mediante la función *classification_report*.

4.3.1. Confusion Matrix

Confusion Matrix significa matriz de confusión, y permite ver la clasificación de los resultados obtenidos a partir de modelo de predicción utilizado. La matriz tiene más columnas y filas según las categorías que tengan las *labels* de los datos. Las columnas indican el resultado obtenido, y en cambio las filas el valor que en realidad debería haber predicho el modelo. Lo importante es tener el máximo número en la diagonal de la matriz y el mínimo en los otros espacios. En este proyecto nos interesa solo una celda en concreto de la diagonal de la matriz, la cual queremos maximizar.

Se realiza a continuación de la matriz de confusión para un mayor entendimiento con solo dos categorías:

	Valor predicho categoría baja	**Valor predicho categoría alta**
Valor real para la categoría baja	Valores predichos como categoría baja y tienen un valor real como categoría baja [AB]	Valores predichos como categoría alta y tienen un valor real como categoría baja [FA]
Valor real para la categoría alta	Valores predichos como categoría baja y tienen un valor real como categoría alta [FB]	Valores predichos como categoría alta y tienen un valor real como categoría alta [AA]

Tabla 1. Descripción detallada de Matriz de confusión

4.3.2. Precision

Este método se utiliza para poder medir la calidad del modelo utilizado. Se calcula para cada grupo individualmente. Es muy eficaz ya que divide el total de aciertos para una categoría en concreto entre la suma de los aciertos y los fallos en esa categoría. La formula es la descrita a continuación:

$$Precision = \frac{AA}{AA + FA}$$

4.3.3. *Recall*

Recall informa sobre la cantidad de datos que el modelo puede identificar. Este método también se utiliza solo para una categoría individualmente. Divide los aciertos para una categoría en concreto entre el número total que tendría que haber de esa categoría.

$$Recall = \frac{AA}{AA + FB}$$

4.3.4. F1-score

F1-score es el método que se utiliza en este proyecto para saber cuales son los mejores parámetros del modelo escogido. Es una combinación de *Recall* y *precisión* de una categoría en concreto, ya que hace la media harmónica obteniendo un solo valor. Es muy útil debido a que tiene en cuenta las dos métricas, de manera que al aumentar su valor las dos métricas también deben de haber aumentado considerablemente. Si el modelo tiene un gran valor de F1, significa tener un gran valor de *Recall* y *precisión*, lo que conlleva a manejar bien la categoría objetivo.

$$F1 - score = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

4.3.5. Accuracy score

Accuracy score es una métrica de evaluación el cual, mediante una función, se puede obtener de los modelos utilizados. Mide el porcentaje de casos que el modelo ha acertado con su predicción. Es una de las métricas más usadas para analizar los datos obtenidos. En este proyecto el inconveniente es que tiene en cuenta los fallos de todas las categorías que tengan los datos. Accuracy score puede causar confusiones, al dar un buen porcentaje y pensar que se están haciendo buenas predicciones, pero en realidad que el modelo esté desbalanceado, es decir, que prediga mejor unas categorías que otras.

Para obtener este porcentaje utiliza la formula que se muestra seguidamente:

$$Accuracy\ score = \frac{AB + AA}{AB + AA + FB + FA}$$

5. Análisis de resultados

Una vez presentadas las herramientas, los modelos y las métricas usadas en el proyecto, se procedió a crear los diferentes experimentos. En estos experimentos se ha utilizado, como en apartados anteriores se ha comentado, la validación *Holdout*, los datos están bien distribuidos gracias al uso de una función que aporta este tipo de validación. En los diferentes experimentos se utilizaron los datos de dos categorías ya que se consideraron para óptimos al no disponer de todo el tiempo deseado.

A continuación, se hace una breve explicación de los experimentos realizados. En el primer experimento se creó un modelo de *Regresión Logística* con los datos de dos categorías, este tipo de modelo es uno de los más sencillos de realizar. El valor de *F1-score* logrado a partir de este modelo se utilizó como valor base para superarlo con el modelo de *Random Forest*. Este se comparó seguidamente con el modelo *Random Forest* creado a partir de los mismos datos dejando todos sus parámetros por defecto.

A partir de saber cuales eran las variables más importantes, se consideró añadir columnas nuevas al modelo para ver si así se lograr mejores predicciones mediante el modelo de *Random Forest* dejando los parámetros de *Random Forest* y los de los árboles de decisión.

Una vez escogidos los datos a utilizar se realizaron una serie de experimentos, donde en unos se modificaban primero los parámetros de *Random Forest* y después los parámetros de los árboles de decisión y otros de forma inversa, primero se modificaban los parámetros de los árboles de decisión y luego los del *Random Forest*. Los resultados logrados de cada experimento se comentaron para poder realizar una comparativa óptima.

5.1. Resultado Base

En el modelo construido de *Random Forest* en este apartado, tanto los parámetros de *Random Forest* como los de los árboles de decisión no se modificaron, estaban por defecto. El resultado obtenido de la métrica de evaluación *F1-score* de *Regresión Logística* de la categoría popularidad alta, grupo 1, se utilizó para compararlo con los resultados de los modelos de *Random Forest* a medida que se fueron mejorando. Se hizo énfasis en superar este valor de *F1-score*.

	precision	recall	f1-score	support
0	0.63	0.59	0.61	5547
1	0.66	0.70	0.68	6347
accuracy			0.65	11894
macro avg	0.65	0.65	0.65	11894
weighted avg	0.65	0.65	0.65	11894

Tabla 2. Métricas de evaluación para datos con dos categorías con RL

	precision	recall	f1-score	support
0	0.59	0.65	0.62	5547
1	0.66	0.60	0.63	6347
accuracy			0.62	11894
macro avg	0.62	0.62	0.62	11894
weighted avg	0.63	0.62	0.62	11894

Tabla 3. Métricas de evaluación para datos con dos categorías con RF

El valor exacto de *F1-score* conseguido con el modelo de *Regresión Logística* fue 0.682. Para este experimento se puede decir que el modelo se ajustaba bien a los datos. Era un modelo que había generalizado de una forma bastante eficiente.

Una vez obtenidos los valores de *F1-score* del modelo *Random Forest*, se puede decir que el modelo para esta categoría tenía fallos, pero no era mal modelo. Exactamente el modelo obtuvo un valor de 0.628 de *F1-Score*. En los datos con dos categorías, era importante que las dos tuvieran la misma cantidad de datos con el fin de evitar un sobreajuste del modelo con una categoría específica. En este caso, el modelo se ajusta de manera semejante a diferencia de los resultados proporcionados por el modelo de Regresión Logística.

En comparación, el valor que el modelo *Random Forest* nos proporcionó, no superaba el valor de *F1-score* de el modelo de *Regresión Logística*. Es de esperar que *Random Forest*

se ajuste mejor a los datos si se modifican tanto los parámetros de *Random Forest* como los de los árboles de decisión y no dejarlos por defecto.

Las métricas de evaluación conseguidas de cada uno de los modelos fueron calculadas a partir las dos matrices de confusion que se muestran seguidamente.

	0	1
0	3256	2291
1	1873	4474

Tabla 4. Tabla con las predicciones realizadas para datos con dos categorías con RL

	0	1
0	3596	1951
1	2543	3804

Tabla 5. Tabla con las predicciones realizadas para datos con dos categorías con RF

5.2. Nuevas columnas

Mediante una clase del modelo *Random Forest* usado en el experimento anterior, se pudieron identificar las variables que más influencia tenían el modelo. La clase es llamada *feature_importances_*. A partir de los valores proporcionados se ha realizado un gráfico de barras con el objetivo de poder ver cual era el más significativo respecto a los otros.

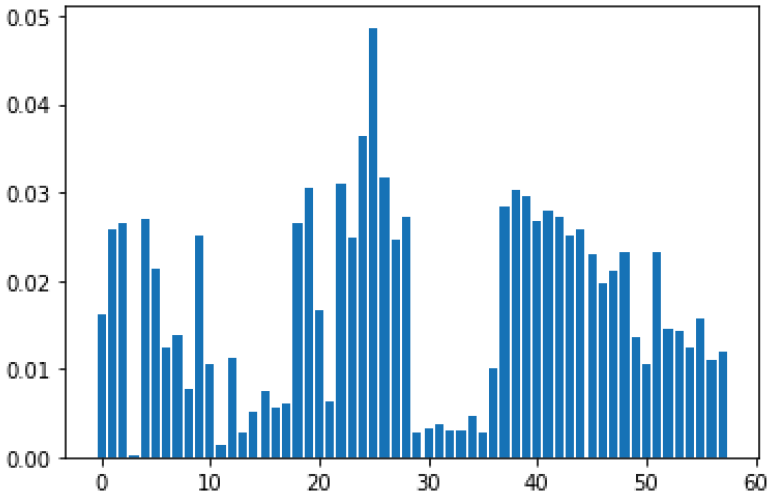

Figura 5. Gráfico donde se representa la importancia de variables para datos con dos categorías

A partir de estos datos sobre cada variable, se procedió a crear tres columnas nuevas para ver si podrían mejorar el modelo creado anteriormente. Estas tres columnas eran una combinación de dos variables diferentes. Esta combinación se hizo mediante la media harmónica de las dos variables. Se escogió realizar la media harmónica para que las dos variables tuvieran la misma influencia en la creación de la columna. Cabe destacar que *F1-score* utiliza la misma formula para combinar las dos variables y es muy fiable. Las variables que se combinaron fueron *n_tokens_content* con *num_imgs* convirtiéndose en la columna número 58, *num_videos* con *n_tokens_title* conviertiendose en la columna 59 y *kw_avg_avg* con *kw_max_avg* dando lugar a la columna número 60. Hay variables que no son tan importantes en el modelo, pero se tuvieron en cuenta para ver como el modelo podía mejorar añadiendo tanto variables significativas como no.

Además, al crear estas nuevas columnas se buscó si había valores *missing* o valores infinitos. En el caso que se encontrara algún valor se sustituía por 0 ya que se consideraba que no había relación y era de poca información.

Por otra parte, se creó un modelo a partir de los datos modificados, dejando los parámetros del *Random Forest* por defecto. Se decidió realizar el modelo igual que el anterior con el objetivo de efectuar una buena comparativa entre los dos modelos.

	precision	recall	f1-score	support
0	0.59	0.65	0.62	5547
1	0.66	0.61	0.63	6347
accuracy			0.63	11894
macro avg	0.63	0.63	0.63	11894
weighted avg	0.63	0.63	0.63	11894

Tabla 6. Métricas de evaluación para datos con dos categorías modificados con RF

	0	1
0	3588	1959
1	2493	3854

Tabla 7. Tabla con las predicciones realizadas para datos con dos categorías modificados con RF

Los resultados conseguidos del modelo construido proporcionaron un valor de *F1-score* de 0.633. Al aumentar el valor de *Recall* también aumentó el valor de *F1-score*, no tanto ya que el valor de *Precision* se mantuvo bastante constante. Se puede decir que las columnas añadidas al modelo, provocaron una mejora en los resultados que se habían obtenido de las predicciones. De este modo se decidió utilizar este tipo de datos de dos categorías para las partes siguientes del proyecto. Se comprobó como afectaban al modelo la creación de estas nuevas columnas mediante la clase descrita anteriormente.

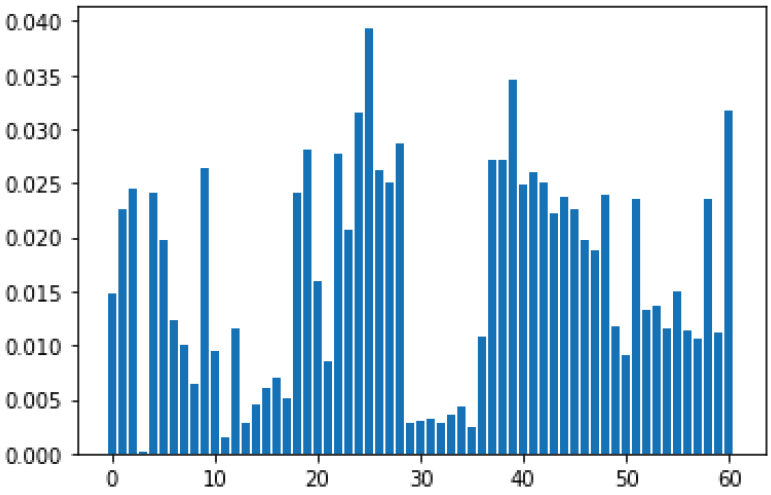

Figura 6. Gráfico donde se representa la importancia de variables para datos con dos categorías modificados

Se puede interpretar como la columna 58 y 60 aportan bastante información para que el modelo haga la predicción. Específicamente la columna 60, al ser la combinación de las dos variables más significativas, obtiene un gran valor, siendo la tercera columna más importante para el modelo.

El valor obtenido aumentó, no obstante, no se consiguió sobrepasar el valor logrado mediante el modelo de *Regresión Logística.*

5.3. Primera fase: Modificación parámetros del AD

En este experimento, una vez definidos los datos a utilizar, se decidió obtener los valores de *F1-score* variando solo los parámetros de los árboles de decisión y dejando por defecto los parámetros del *Random Forest*. Los parámetros de los árboles que se consideraron de mayor relevancia fueron *Max_depth* y *Min_samples_leaf.* A continuación, se muestra un gráfico donde se pueden observar los valores obtenidos de *F1-score* para cada combinación de estos dos parámetros.

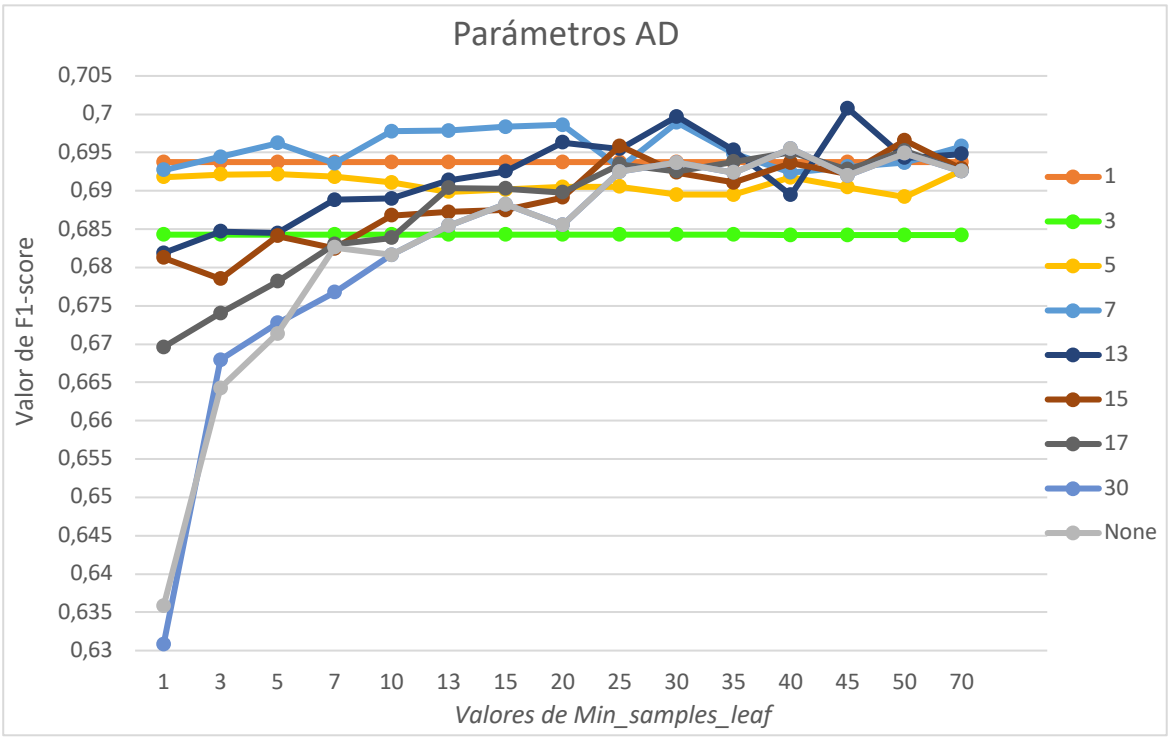

Figura 7. Gráfico donde se representan diversos valores de F1-score para parámetros AD

El eje de las *X* representa los valores de *Min_sample_leaf* que se fueron imponiendo en el modelo. Cada una de las rectas determina el valor de *Max_depth* que se utilizó, estas rectas se pueden diferenciar mediante colores. El eje de las *Y* representa los valores que se obtuvieron para cada una de las combinaciones.

Se puede observar como para valores grandes de *Max_Depth* y valores pequeños de *Min_samples_leaf* el modelo proporciona valores bajos de *F1-score*. En cambio, se obtienen buenos resultados con valores altos de *Min_samples_leaf* y valores de *Max_depth*

no muy grandes. En el grafico no se han utilizado todos modelos creados, si se desea se pueden observar todos en el apartado de anexos. Los que no se han usado es debido a que causaban mucha confusión a la hora de interpretar el gráfico.

Una vez conseguidos estos valores, se decidió escoger los que tenían mayor valor de *F1-score* para seguir con el proyecto. Las combinaciones seleccionadas fueron las siguientes:

- *Max_depth*=13-*Min_samples_leaf*=45→ Valor de *F1-score*= 0.700
- *Max_depth*=7-*Min_samples_leaf*=30→ Valor de *F1-score*= 0.698
- *Max_depth*=15-*Min_samples_leaf*=50→ Valor de *F1-score*= 0.696
- *Max_depth*=14-*Min_samples_leaf*=20→ Valor de *F1-score*= 0.695
- *Max_depth*=20-*Min_samples_leaf*=40→ Valor de *F1-score*= 0.695
- *Max_depth*=17-*Min_samples_leaf*=50→ Valor de *F1-score*= 0.695
- *Max_depth*=1-*Min_samples_leaf*=1→ Valor de *F1-score*= 0.693

5.4. Primera fase: Modificación parámetros *RF*

Una vez seleccionadas las combinaciones, se procedió a modificar los parámetros del *Random Forest*. Los parámetros que se modificaron eran *N_estimator* y *Max_features.* Se crearon un total de 7 experimentos, donde cada uno de los dos parámetros del árbol de decisión se fijaban con una de las combinaciones, y se modificaban los dos parámetros del *Random Forest*. Estos experimentos mostraban cuál sería la mejor combinación de parámetros para conseguir buenos resultados. Además, se puede observar que todas las combinaciones escogidas superan el valor obtenido con el modelo de *Regresión Logística.*

A continuación, se muestra las gráficas de cada experimento realizado con sus valores.

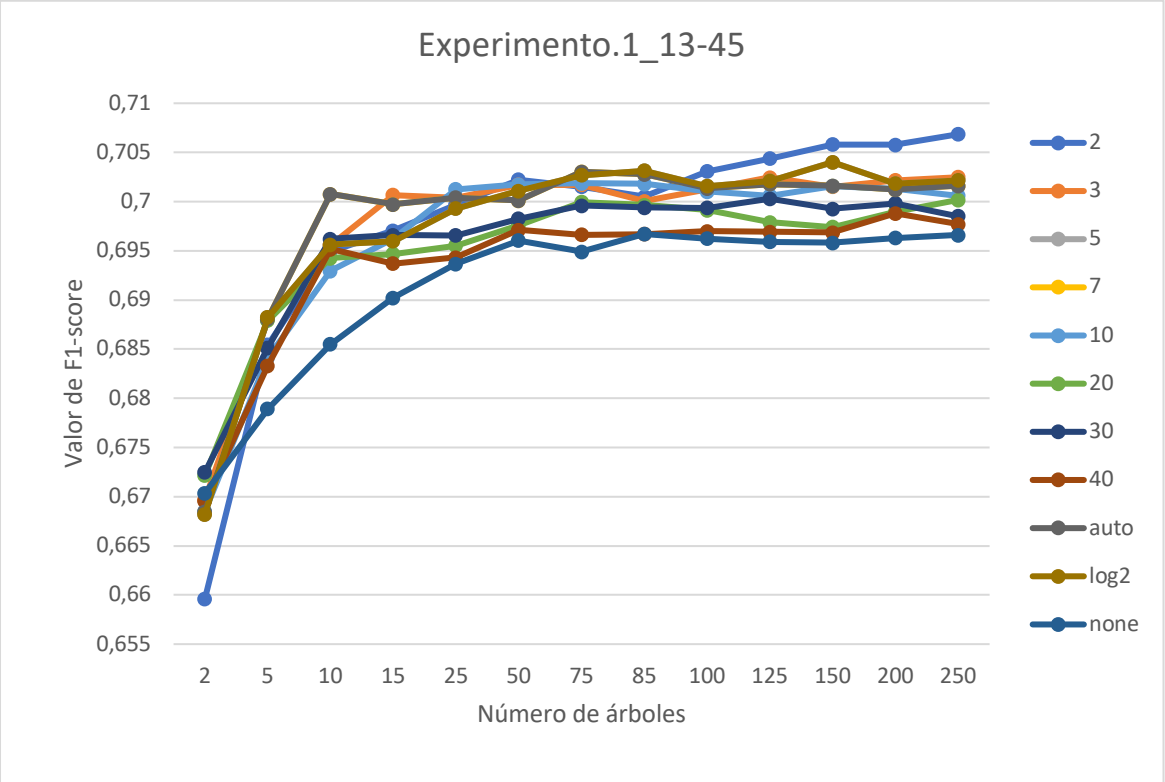

Figura 8. Gráfico donde se muestran los valores obtenidos de F1-score para el exp.1

En este experimento los valores obtenidos de *F1-score* aumentaron dependiendo del número de árboles que el modelo *Random Forest* creaba. A partir de un valor de 25 aproximadamente, el modelo se estabilizaba y se observaban valores similares. En el caso

de utilizar solo 2 variables para crear el árbol, los valores de *F1-score* aumentaban, pero no en exceso en comparación a otro número de árboles de decisión. Se seleccionó los valores obtenidos más altos de *F1-score*, y como la diferencia no era suficientemente alta se procedió a buscar los valores *Recall* y *Precision* correspondientes.

- *Max_features*=2-*N_estimators*=250→*F1-score*=0.706→ *Precision* = 0.652→ *Recall*=0.771
- *Max_features*=2-*N_estimators*=150→*F1-score*=0.705→ *Precision* = 0.651→ *Recall*=0.769
- *Max_features*=2-*N_estimators*=200→*F1-score*=0.705→ *Precision* = 0.651→ *Recall*=0.769

Al utilizarse solo dos variables, se consideró oportuno crear una gráfica con los valores de las variables más significantes para modelos de este tipo.

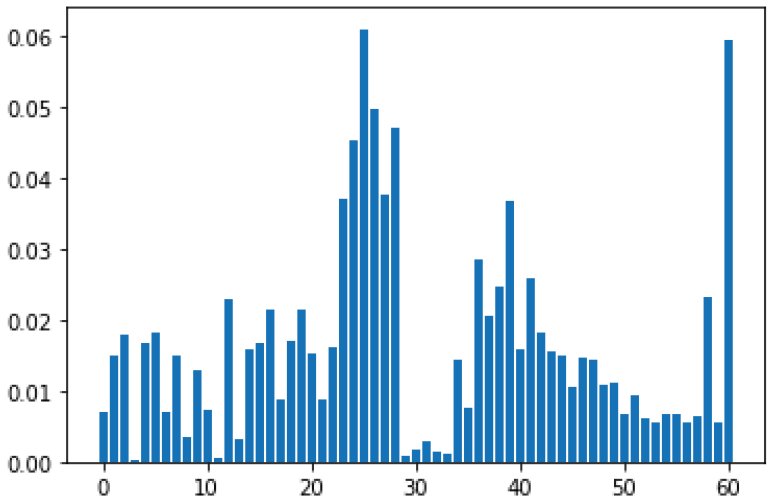

Figura 9. Gráfico donde se representa la importancia de las variables en exp.1

Se puede observar cómo una de las variables más significas es la que se creó y se añadió al modelo, la columna 60. Se puede ver cómo en este caso el grafico sigue la misma tendencia que el obtenido al final del apartado de nuevas columnas.

Se realizó el segundo experimento con la combinación en la que se obtuvo el segundo valor más alto de *F1-score*.

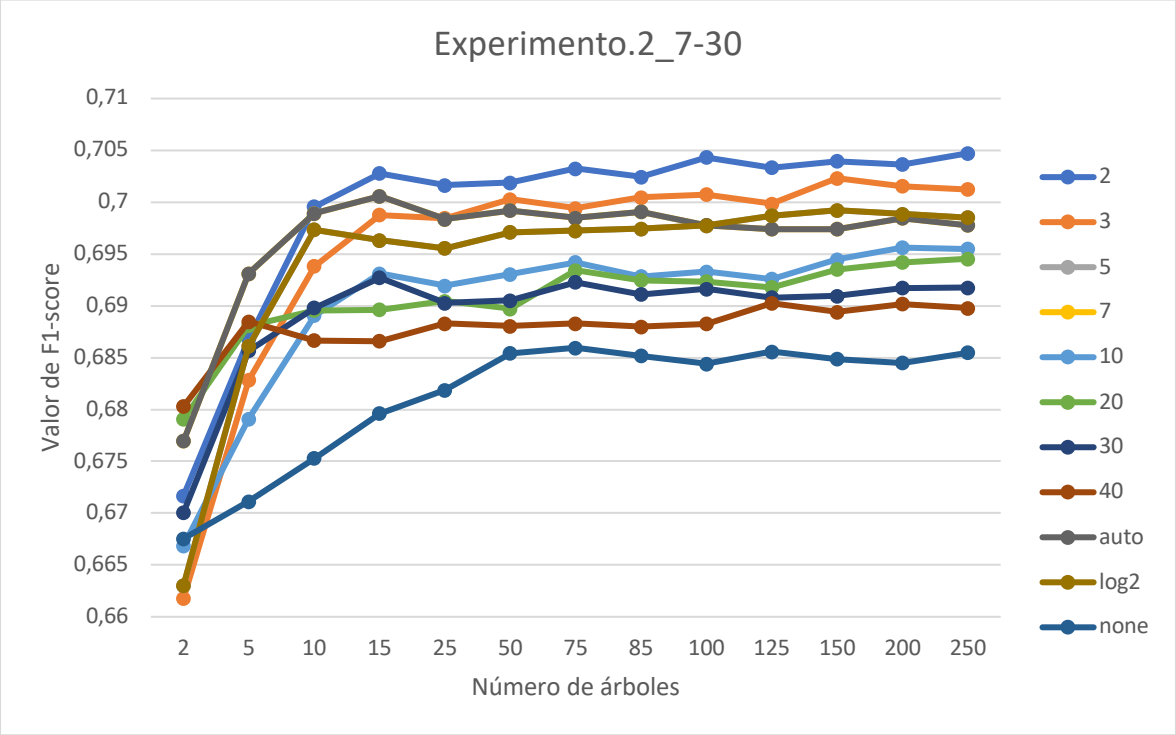

Figura 10. Gráfico donde se muestran los valores obtenidos de F1-score para el exp.2

En este experimento se observa cómo en las rectas obtenidas hay una pequeña separación, a diferencia del experimento anterior. Se puede diferenciar claramente cada recta sin estar unas sobre las otras, es decir, en pocas ocasiones se obtuvieron los mismos valores de *F1-score*. Respecto al parámetro de *N_estimators*, a un mayor valor se obtuvieron buenos resultados y con un valor pequeño, los valores *F1-score* no eran lo suficientemente altos para superar el valor aportado por el modelo de *Regresión Logística.* Los valores más altos de *F1-score* eran los conseguidos a partir de un valor del parámetro *Max-Features* pequeño y un valor alto de *N_estimators*, igual que ocurría el en el experimento anterior.

Se seleccionaron los tres valores más altos, para ver si podía haber una diferencia mayor se muestran los valores de las métricas *Recall* y *Precision*.

- *Max_features*=2-*N_estimators*=250→*F1-score*=0.704→ *Precision*= 0.642→ *Recall*=0.780
- *Max_features*=2-*N_estimators*=100→*F1-score*=0.704→ *Precision*= 0.642→ *Recall*=0.779

- *Max_features*=2-*N_estimators*=150→*F1-score*=0.703→ *Precision*= 0.642→ *Recall*=0.778

En este experimento, igual que el anterior, una de las variables más significas para conseguir buenas predicciones era la última que se creó y fue añadida a los datos.

Se procedió a realizar el tercer experimento con el tercer valor más alto, a partir de las combinaciones descritas anteriormente.

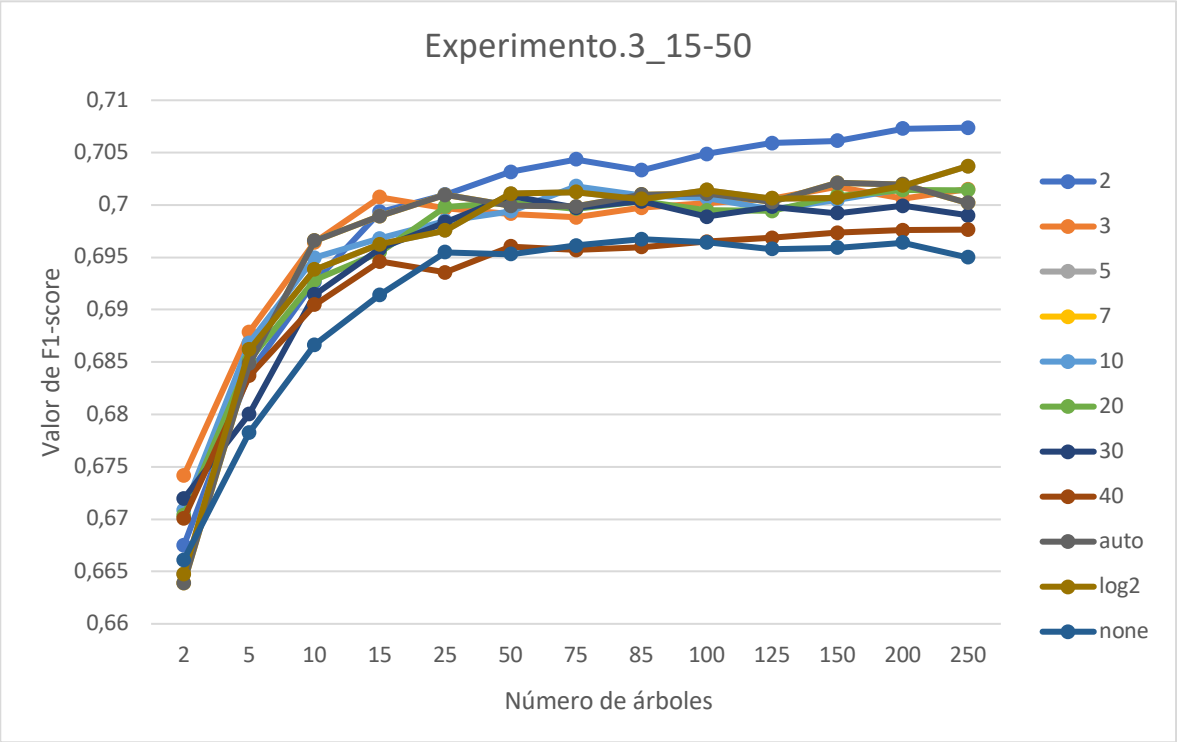

Figura 11. Gráfico donde se muestran los valores de F1-score obtenidos para el exp.3

En este caso, para un valor de 2 del parámetro *Max_features*, los modelos proporcionaban un valor bastante alto de *F1-score*. A partir de 50 árboles de decisión, se observa claramente que la recta de valor 2 de *Max_features* aumentó más que las otras rectas obtenidas. Para un número pequeño de árboles el modelo realizaba peores predicciones, siendo los valores de *F1-score* inferiores que los aportados por el modelo de *Regresión Logística*.

Tal y como se realizó en los experimentos anteriores, se buscaron los valores de las métricas *Recall* y *Precision* para los tres valores más altos.

- *Max_features*=2-*N_estimators*=250→*F1-score*=0.707→ *Precision*= 0.654→ *Recall*= 0.770
- *Max_features*=2-*N_estimators*=200→*F1-score*=0.707→ *Precision*= 0.654→ *Recall*= 0.769
- *Max_features*=2-*N_estimators*=150→*F1-score*=0.706→ *Precision*= 0.653→ *Recall*= 0.768

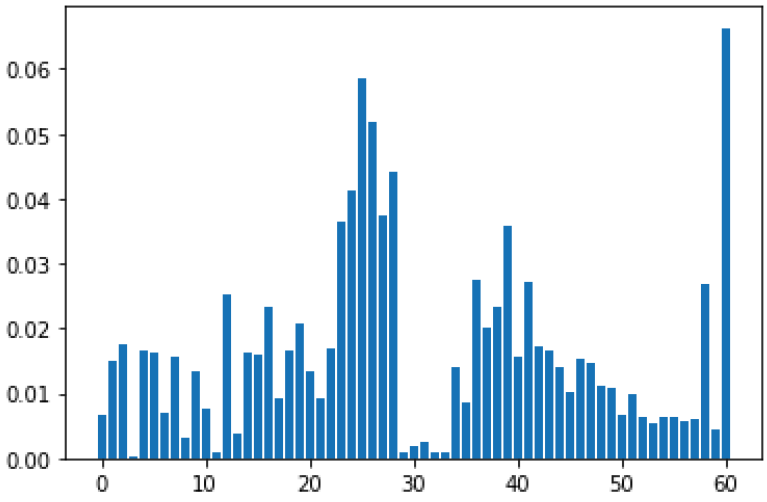

Figura 12. Gráfico donde se representa la importancia de las variables en exp.3

La columna añadida a los datos en este experimento es la más significativa. En los otros casos para el modelo también era importante, pero no la más destacada.

Se decidió realizar 4 experimentos con las combinaciones restantes para poder observar la manera en que *Random Forest* actuaba en cada uno de ellos.

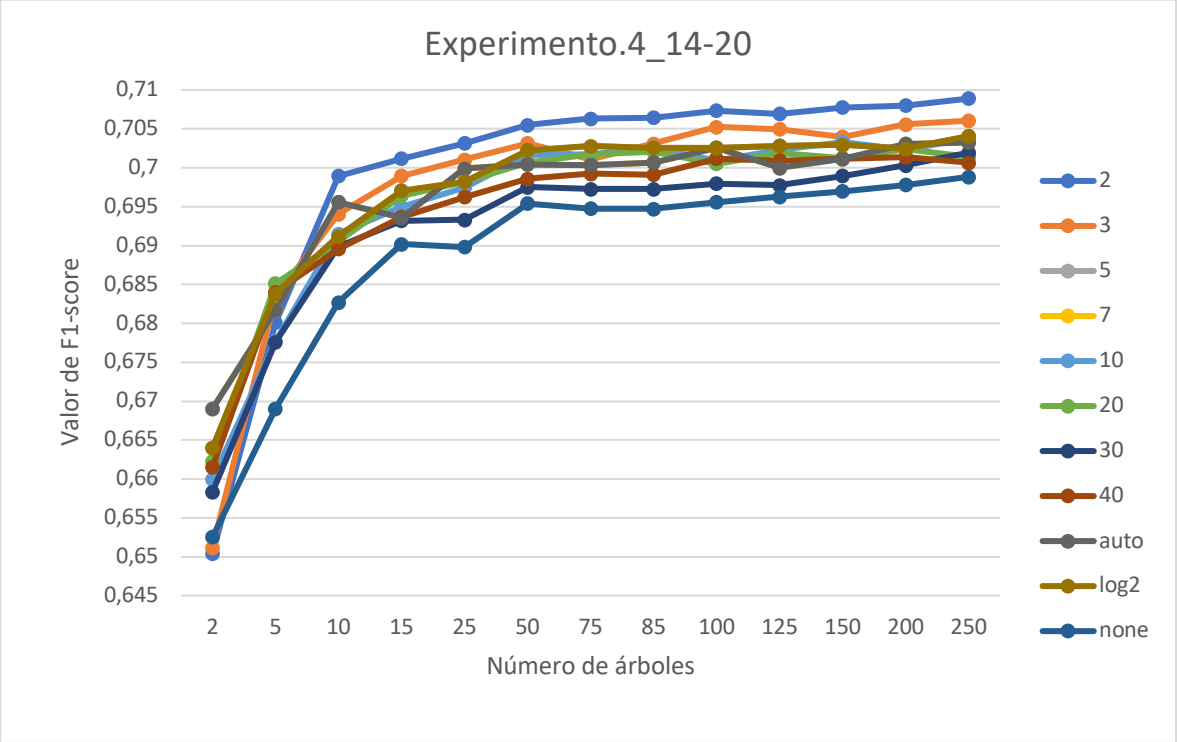

Figura 13. Gráfico donde se muestran los valores de F1-score obtenidos para el exp.4

En este experimento, si se escogen todas las variables para crear el modelo, las predicciones consiguen unos valores de *F1-score* bastante bajos. A partir de 15 árboles de decisión, se lograron valores que superaban las métricas resultantes del modelo de *Regresión Logística*. En todo caso, se conseguía el mayor valor de *F1-score* con un gran número de árboles y un valor bajo de *Max_features*.

A continuación, se muestran los valores *Recall y Precision* de los tres mejores valores de *F1-score*.

- *Max_features*=2-*N_estimators*=250→*F1-score*=0.708→ *Precision*= 0.658→ *Recall*= 0.766
- *Max_features*=2-*N_estimators*=200→*F1-score*=0.708→ *Precision*= 0.658→ *Recall*= 0.766
- *Max_features*=2-*N_estimators*=150→*F1-score*=0.707→ *Precision*= 0.657→ *Recall*= 0.766

En este experimento ocurre lo mismo que en el experimento 3 donde la variable más significativa para el modelo es la creada y añadida al principio.

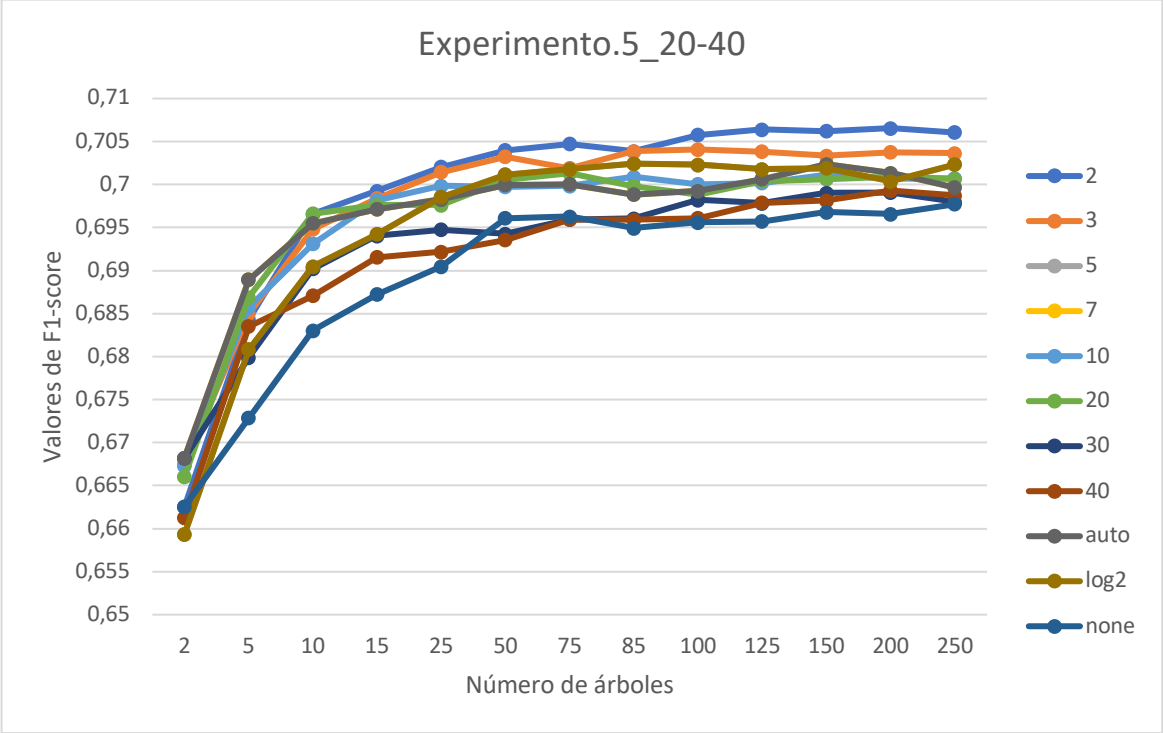

Figura 14. Gráfico donde se muestran los valores de F1-score obtenidos para el exp.5

A partir el gráfico creado se observa cómo el modelo aumentó los valores de *F1-score* a medida que se incrementaba el número de árboles de decisión a crear por el modelo. Respecto a *Max_features*, las rectas seguían una misma tendencia, donde destaca la de valor 2 pero no logró valores muy mayores a la de valor 3. En todo caso, haría falta utilizar como mínimo 50 árboles para superar el valor *F1-score* conseguido en el modelo de *Regresión Logística*.

A continuación, se determinaron los valores de las métricas de evaluación *Recall* y *Precision* para los tres valores más altos en este tipo de experimento.

- *Max_features*=2-*N_estimators*=200→*F1-score*=0.706→ *Precision*= 0.655→ *Recall*= 0.766
- *Max_features*=2-*N_estimators*=125→*F1-score*=0.706→ *Precision*= 0.655→ *Recall*= 0.765
- *Max_features*=2-*N_estimators*=150→*F1-score*=0.706→ *Precision*= 0.655→ *Recall*= 0.765

Igual que el experimento anterior, la variable dominante en el modelo es la columna añadida. Consigue un valor muy alto ya que es la combinación de la segunda y tercera variable más dominante del modelo.

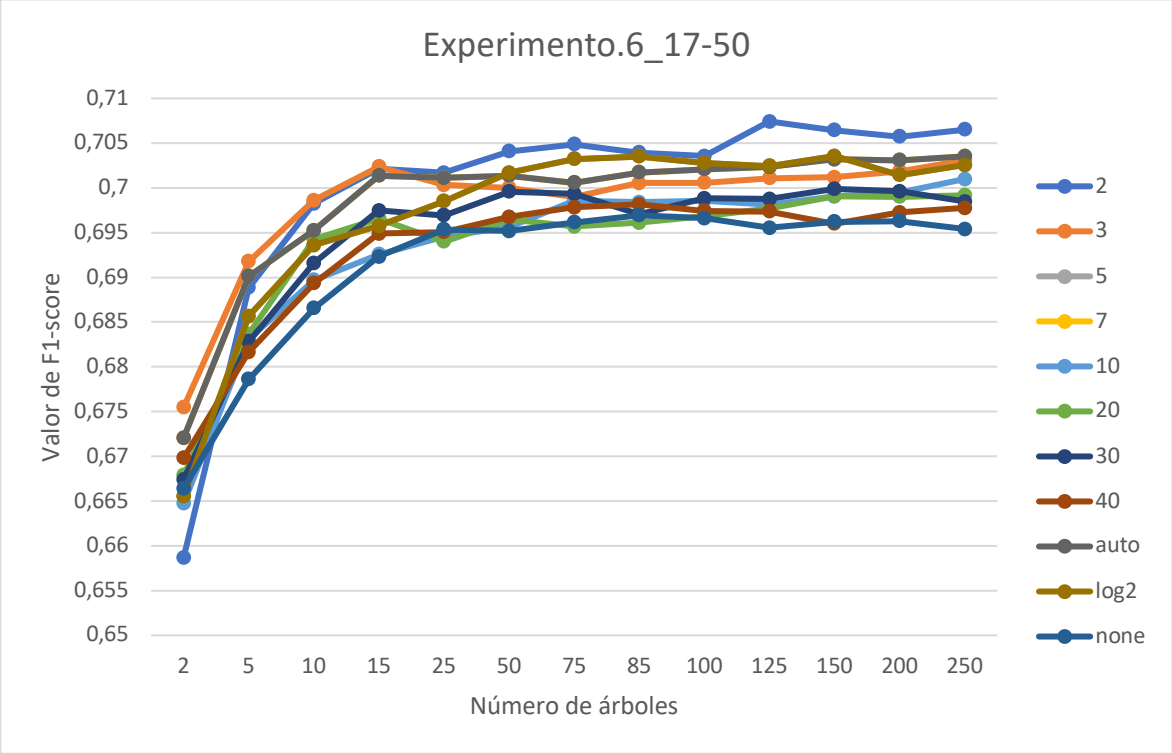

Figura 15. Gráfico donde se muestran los valores de F1-score obtenidos para el exp.6

Se obtuvieron los valores máximos de *F1-score* imponiendo un valor 2 al parámetro *Max_features* y creando un gran número de árboles. No obstante, si no se creaba un gran número de árboles con un valor 2 de *Max_features,* el modelo no generalizaba correctamente y se observaban valores bastante inferiores, no superando el valor de *F1-score* del modelo de *Regresión Logística*. Por otra parte, los resultados seguían siempre la misma tendencia, y si se utilizaban todas las variables para crear los árboles y se creaban un gran número de árboles, el modelo no generalizaba lo suficiente.

Siguiendo la misma metodología que los otros experimentos, se seleccionaron las combinaciones más eficientes del experimento y se consiguieron los valores *Recall* y *Precision* de cada una.

- *Max_features*=2-*N_estimators*=125→*F1-score*=0.707→ Precision= 0.653→ *Recall*= 0.772

- *Max_features*=2-*N_estimators*=250→*F1-score*=0.706→ *Precision*= 0.651→ *Recall*= 0.771
- *Max_features*=2-*N_estimators*=150→*F1-score*=0.706→ *Precision*=0.652→ *Recall*= 0.770

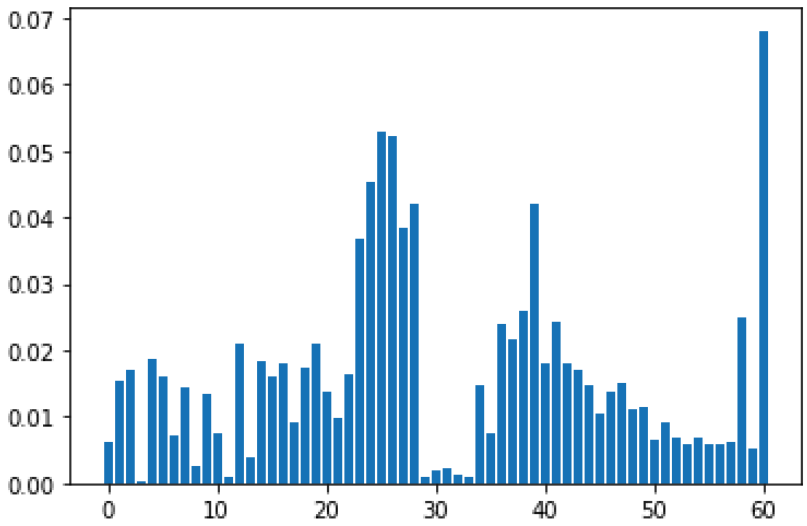

Figura 16. Gráfico donde se representa la importancia de las variables en exp.6

En este experimento se observa como la variable 26, *self_reference_min_shares,* aporta mucha información al modelo para sus predicciones; logrando un valor semejante a la segunda variable más importante, *Kw_avg_avg.* No obstante, la variable más importante es la creada de la combinación descrita anteriormente.

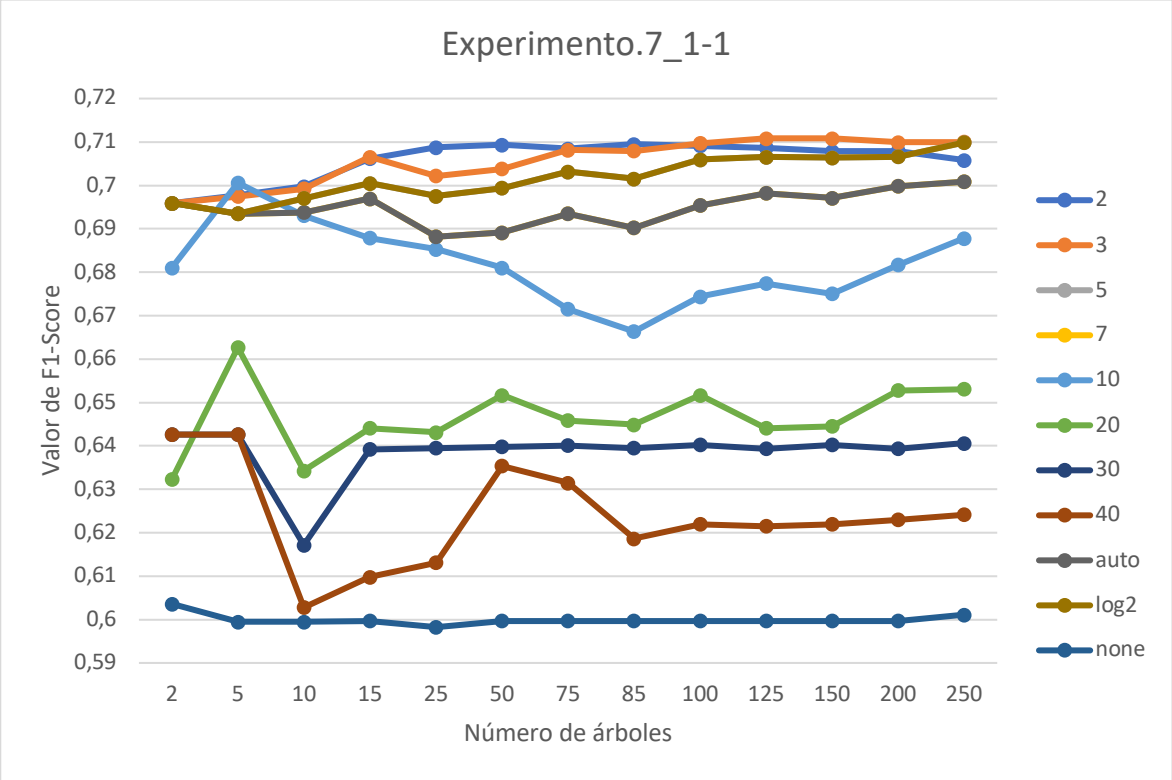

Figura 17. Gráfico donde se muestran los valores de F1-score obtenidos para el exp.7

En este experimento, se encuentra una tendencia diferente de las rectas, no vista en las combinaciones anteriores. Normalmente los valores de *F1-score* iban aumentando a medida que el número de árboles se incrementaba, pero en este caso se puede ver como los valores disminuyen o incrementan poco. Especialmente cuando se impusieron valores de 10 y 40 en el parámetro *Max_features*, al ir aumentando el número de árboles, se observó un gran decrecimiento del valor de *F1-score*, pero posteriormente se produjo un crecimiento.

Las rectas en otros experimentos solían estar bastante juntas, cosa que se interpreta como valores bastante similares, pero en esta situación los modelos creados proporcionaron valores muy distintos. Estos valores variaban en función del parámetro *Max_features*, y al imponer valores altos de este, el modelo empeoró. A partir de un valor de 10, la diferencia era considerable.

Con valores pequeños de *Max_features* se obtuvo el mejor rendimiento del modelo, donde los valores eran bastante iguales.

A continuación, se muestran las métricas *Precision* y *Recall* de los resultados de los modelos que han generalizado mejor.

- *Max_features*=3-*N_estimators*=150→*F1-score*=0.710→ *Precision*= 0.570 → *Recall*= 0.942
- *Max_features*=3-*N_estimators*=125→*F1-score*=0.710→ *Precision*= 0.574→ *Recall*= 0.932
- *Max_features*=3-*N_estimators*=100→*F1-score*=0.709→ *Precision*= 0.575→ *Recall*= 0.926

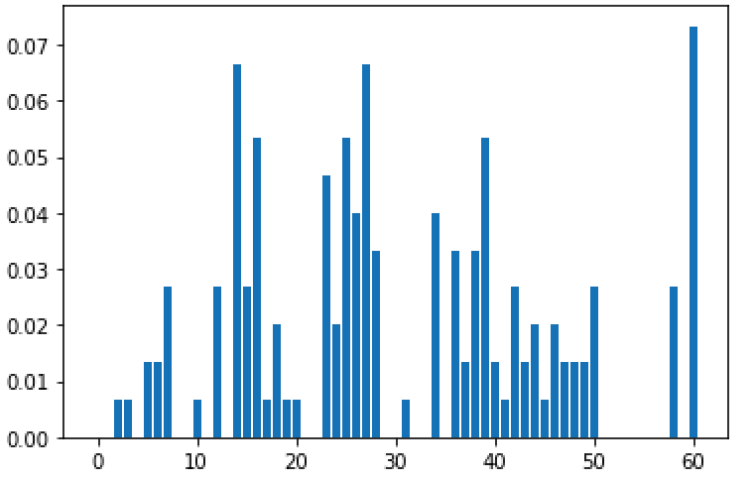

Figura 18. Gráfico donde se representa la importancia de las variables en exp.7

A partir de esta combinación de parámetros se buscó la variable que el modelo tomaba más en consideración. Se puede ver como la más importante es la misma que en los otros experimentos. A diferencia de los otros casos, hay bastantes variables que lograr tener una gran importancia en el modelo, obteniendo valores bastante similares.

5.5. Segunda fase: Modificación parámetros de *RF*

Se planteó la idea de poder mejorar el modelo de *Random Forest*, realizando los experimentos realizados en apartados anteriores, pero de forma inversa: se buscó los mejores valores de los parámetros del *Random Forest* los cuales proporcionaban los valores más altos de *F1-score*. Dejando los otros parámetros del modelo por defecto.

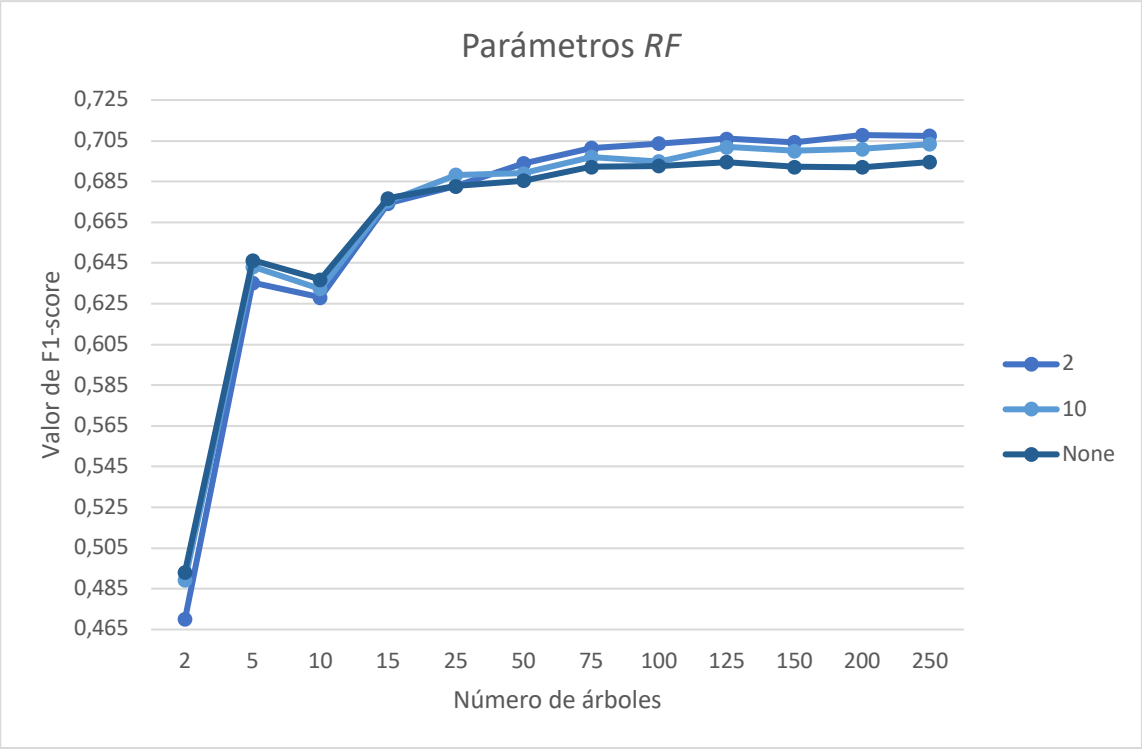

Figura 19. Gráfico donde se representan diversos valores de F1-score para parámetros RF

El eje de las *X* representa el número de árboles que se han utilizado. Por otro lado, el eje de las *Y* representa los valores de *F1-score* obtenidos de los modelos. Las distintas rectas muestran los valores de *Max_features* que se han escogido.

En este experimento, solo se han graficado tres valores de *Max_features* ya que los datos obtenidos son muy semejantes y se crean rectas una sobre otra. Los datos a partir de donde se ha creado la grafica se muestran en los anexos.

Respecto al otro parámetro, a medida que se fue aumentando el número de árboles, los modelos mejoraban. El valor *F1-score* sufrió un pequeño descenso con 10 árboles de decisión, pero luego crece de nuevo. Al llegar al valor de 75 árboles de decisión, los modelos obtienen valores bastante constantes.

Se seleccionaron las mejores combinaciones de estos dos parámetros para seguir realizando experimentos variando los parámetros del árbol. Las mejores combinaciones eran aquellas en las que se imponía un valor de 2 del parámetro *Max_features*.

- *Max_features*=2-*N_estimators*=200→ Valor de *F1-score*= 0.707
- *Max_features*=2-*N_estimators*=250→ Valor de *F1-score*= 0.707
- *Max_features*=2-*N_estimators*=125→ Valor de *F1-score*= 0.706
- *Max_features*=2-*N_estimators*=150→ Valor de *F1-score*= 0.704
- *Max_features*=7-*N_estimators*=125→ Valor de *F1-score*= 0.703
- *Max_features*=2-*N_estimators*=100→ Valor de *F1-score*= 0.703
- *Max_features*=10-*N_estimators*=250→ Valor de *F1-score*= 0.703

5.6. Segunda fase: Modificación parámetros de AD

Una vez obtenidas las mejores combinaciones de parámetros de *Random Forest* se procedió a realizar siete experimentos. En los gráficos que se crearon de cada experimento de este apartado, no se graficaron todas las rectas del parámetro *Max_depth*. En el apartado anexos hay tablas donde muestran todos los datos de cada experimento, se destaca los valores escogidos de color amarillo.

Se realizó el primer experimento de esta parte con la combinación la cual aportó el valor más alto de *F1-score.*

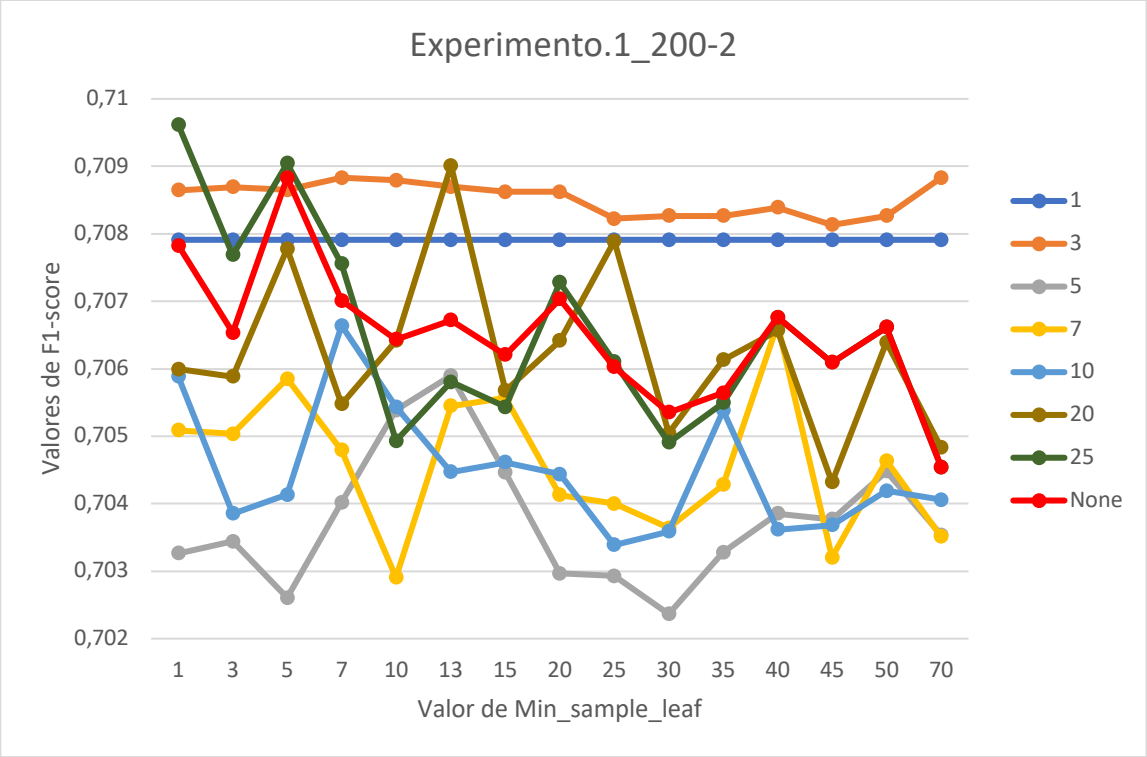

Figura 20. Gráfico donde se muestran los valores de F1-score obtenidos para el exp.1

En este experimento se observa que los valores obtenidos de *F1-score* están bastante dispersos, en pocas situaciones los valores *F1-score* coinciden. Las rectas tienen partes crecientes y decrecientes, donde se pueden identificar claramente sus máximos y mínimos. Cuando se implantó un valor de 1 y 3 en el parámetro *Max_depth*, los valores de *F1-score* se mantenían bastante constantes y, además, eran unos de los casos donde se obtuvieron mejores resultados en las predicciones.

Se extrajeron los resultados más altos de los valores *F1-score* aportados por los diferentes modelos y los valores *Recall* y *Precision* de cada combinación extraída.

- *Max_depth*=25-*Min_samples_leaf* =1→ *F1-score*= 0.709→*Precision*= 0.668→ *Recall*= 0.755
- *Max_depth*=25-*Min_samples_leaf* =5→ *F1-score*= 0.709→*Precision*= 0.667→ *Recall*= 0.756
- *Max_depth*=20-*Min_samples_leaf* =13→ *F1-score*= 0.709→*Precision*= 0.662→ *Recall*= 0.762

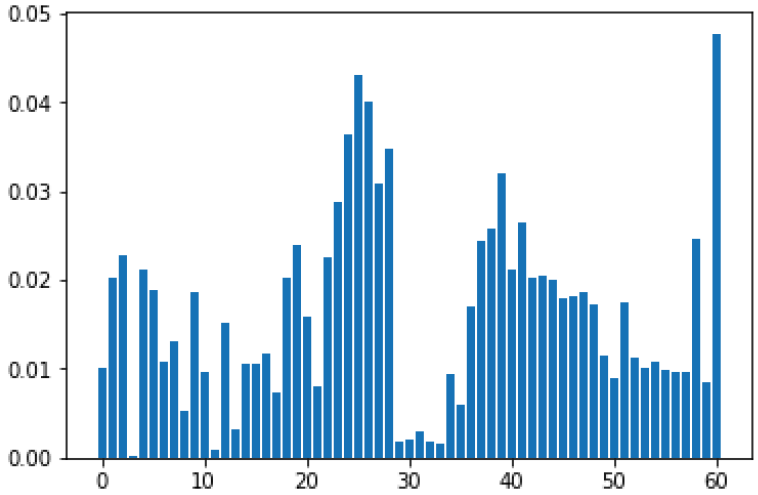

Figura 21. Gráfico donde se representa la importancia de las variables en exp.1

A partir de los modelos se creó un grafico para ver, en este caso, cuáles eran las variables más influyentes en el modelo. La variable más significativa es la columna añadida, combinación de dos variables del modelo.

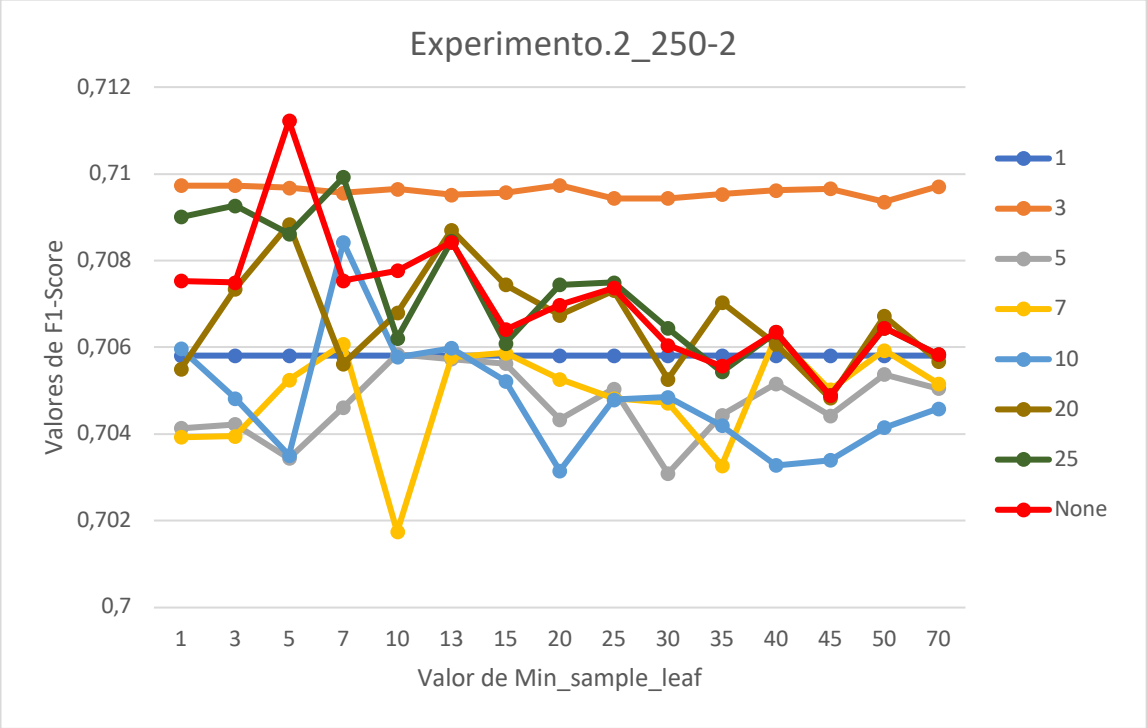

Figura 22. Gráfico donde se muestran los valores de F1-score obtenidos para el exp.2

Se observa como con un valor de 5 en el parámetro *Min_sample_leaf*, en varias ocasiones ocurrió un gran aumento del valor de *F1-score,* donde se lograron los valores más altos para esta combinación. En cambio, con un valor de 10 *Min_sample_leaf*, en varias combinaciones se produjo un decrecimiento bastante alto del valor *F1-score*. Imponiendo un valor de 3 en el parámetro *Max_depth*, se obtiene una recta la cual esta compuesta por valores bastantes similares, sucede lo mismo con el valor 1 pero los valores de F1-score son inferiores.

Se muestra los tres valores más altos con las combinaciones con las que se han conseguido.

- *Max_depth*=None-*Min_samples_leaf* =5→ *F1-score*= 0.711→*Precision*= 0.666→ *Recall*= 0.762
- *Max_depth*=35-*Min_samples_leaf* =5→ *F1-score*= 0.710→*Precision*= 0.665→ *Recall*= 0.761
- *Max_depth*=15-*Min_samples_leaf* =25→ *F1-score*= 0.710→*Precision*= 0.658→ *Recall*= 0.770

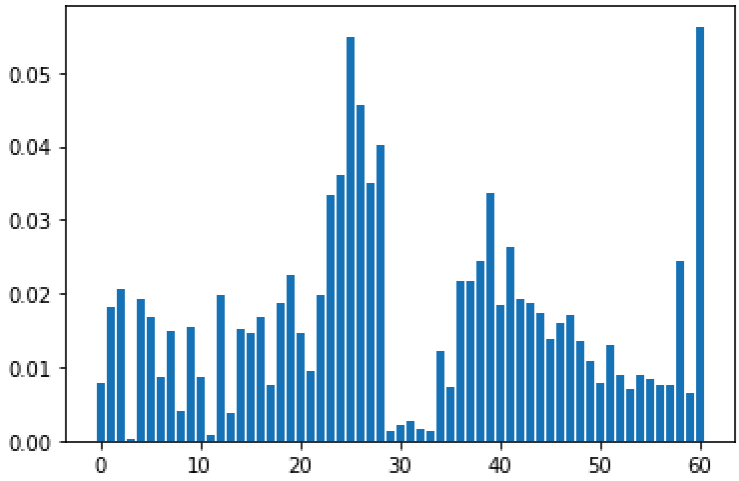

Figura 23. Gráfico donde se representa la importancia de las variables en exp.2

En este caso se observa como los valores obtenidos de dos variables son bastante semejantes, es decir, influyen por igual al modelo a la hora de tomar una decisión. Esto es posible ya que una de ellas es la combinación de la otra con otra variable de menor influencia. Se observa como se mantiene la tendencia de los valores de cada variable.

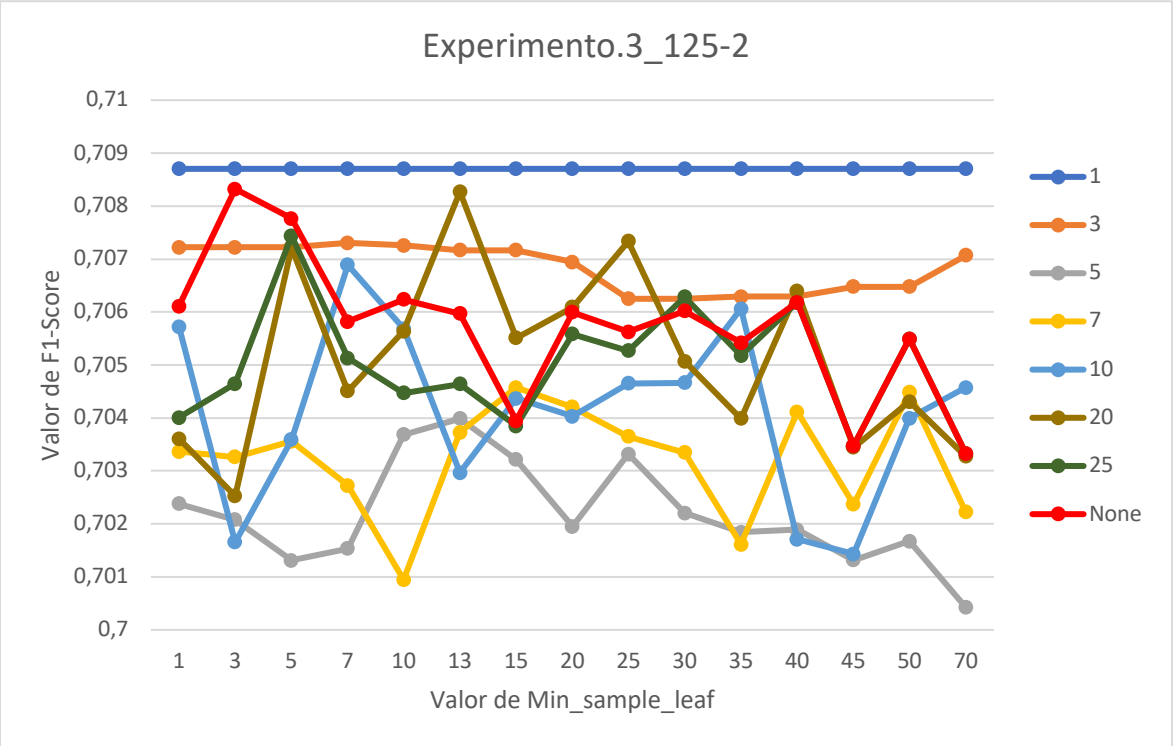

Figura 24. Gráfico donde se muestran los valores de F1-score obtenidos para el exp.3

En este caso se obtuvo el peor valor de *F1-score*, cuando se introdujo un gran valor en el parámetro *Min_sample_leaf* y un valor de 5 en el parámetro *Max_depth*. Para el valor de 5 en *Max_depth* hay combinaciones en las que aumentó el valor, pero acabó disminuyendo notablemente. Por otro lado, con un valor de 15 en *Min_Sample_leaf*, en varias combinaciones se produjo un gran decrecimiento. Con el valor de 1 en *Max_depth*, al modelo no le afectó este suceso, todo lo contrario, la recta creada obtiene valores iguales. Solo hay una combinación que superó el valor constante que logró descrito anteriormente.

A continuación, se muestran los máximos valores de *F1-score* en este experimento, junto de los valores *Precision y Recall* de cada combinación.

- *Max_depth*=35-*Min_samples_leaf* =5→ *F1-score*= 0.708→*Precision*= 0.664→ *Recall*= 0.759
- *Max_depth*=1-*Min_samples_leaf* =1→ *F1-score*= 0.708→*Precision*= 0.558→ *Recall*= 0.968
- *Max_depth*=None-*Min_samples_leaf* =3→ *F1-score*= 0.708→*Precision*= 0.666→ *Recall*= 0.755

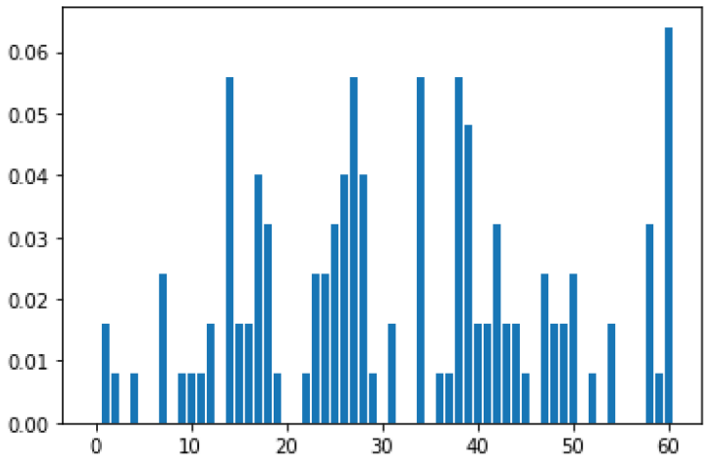

Figura 25. Gráfico donde se representa la importancia de las variables en exp.3

Se puede observar que hay variables las cuales no aportaban información al modelo y por lo tanto se podría abstener de utilizarlas al crear el modelo. Hubo 4 variables que lograron el mismo valor, es decir, tenían la misma importancia para las predicciones del modelo. La columna añadida a los datos, combinación de dos, consiguió un valor superior al 0.06.

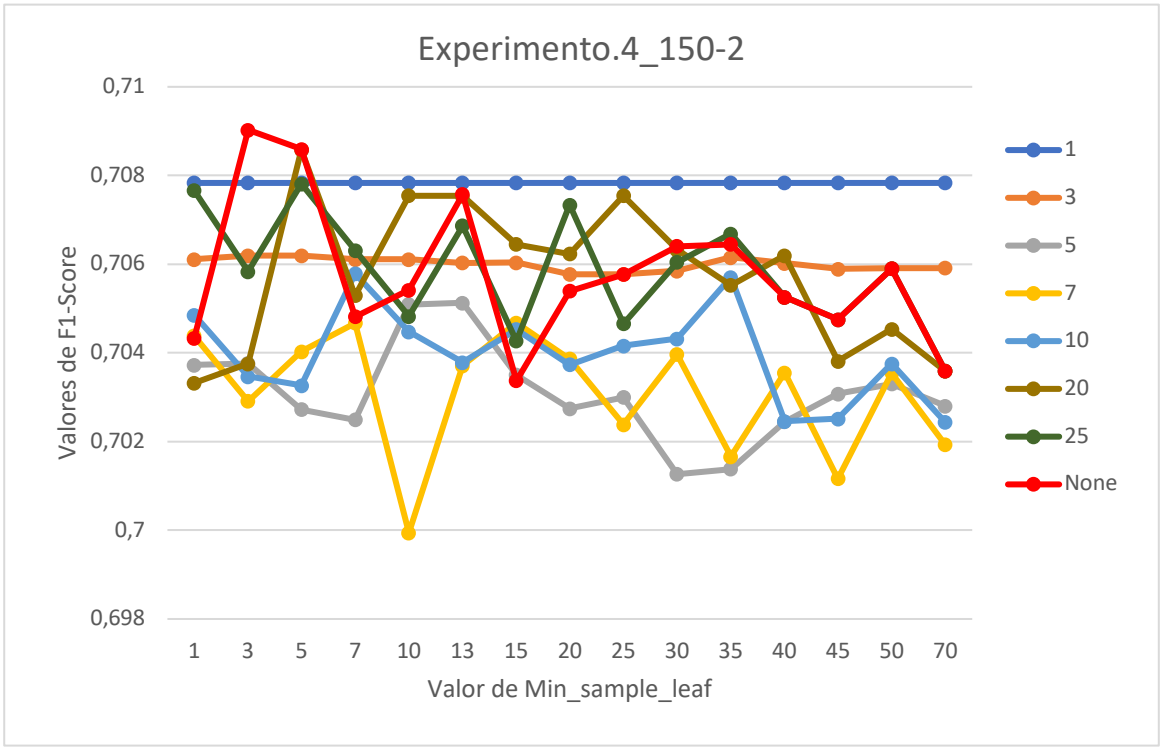

Figura 26. Gráfico donde se muestran los valores de F1-score obtenidos para el exp.4

En este experimento las peores predicciones se obtuvieron cuando se utilizó el valor 7 en el parámetro *Max_depth* y un valor de 10 en *Min_samples_leaf*, en ese punto se produjo un gran decrecimiento del valor de *F1-score*. Por otro lado, seguidamente hubo un aumento, pero no lo suficiente para alcanzar valores altos de *F1-score*. Con la recta de valor 1 *Max_depth,* se observa como los valores se mantuvieron constantes, esto quiere decir que en este caso no influyó nada el parámetro *Min_sample_leaf*. Los máximos valores se lograron con valores altos en *Max_depth* y pequeños en *Min_samples_leaf*.

A continuación, se muestran los valores máximos conseguidos, los valores *Precision* y *Recall* las combinaciones con las que se han logrado.

- *Max_depth*=None-*Min_samples_leaf* =3→ *F1-score*= 0.709→*Precision*= 0.667→ *Recall*= 0.755
- *Max_depth*=35-*Min_samples_leaf* =5→ *F1-score*= 0.708→*Precision*= 0.664→ *Recall*= 0.759

- *Max_depth*=13-*Min_samples_leaf* =20→ *F1-score*= 0.708→*Precision*= 0.658→ *Recall*= 0.766

En este experimento sucedió lo mismo que en el anterior, la variable más significativa fue la última añadida a los datos.

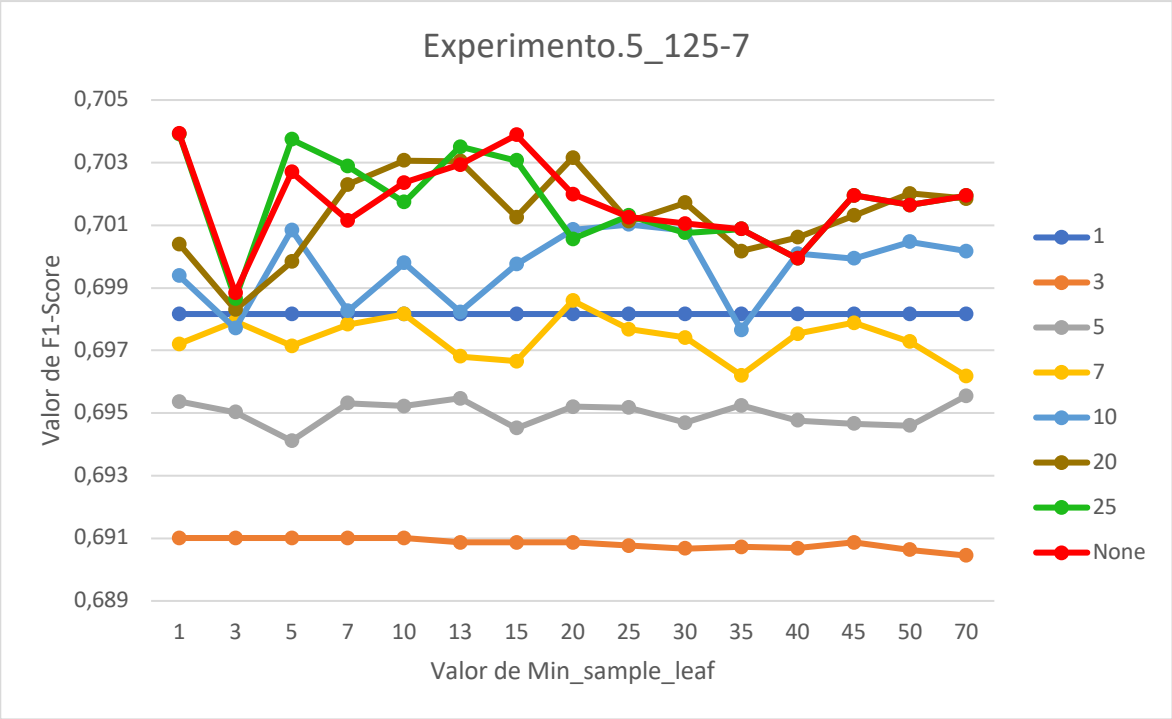

Figura 27. Gráfico donde se muestran los valores de F1-score obtenidos para el exp.5

En este caso se observa como las rectas formadas por valores de *Max_depth* siguen una misma tendencia, esto es debido a que no había mucha influencia de este parámetro. Los valores bajos de *Max_depth* es donde se obtuvieron valores de *F1-score* más pequeños. La diferencia es bastante pequeña ya que los valores obtenidos son bastante iguales. Las peores predicciones se produjeron con el valor de 3 en *Max_depth*. Además, cuando se utilizó el valor de 1 y 3 en *Max_depth*, las rectas se mantienen bastante constantes, significa que el parámetro *Min_samples_leaf* no influyó a la hora de crear el modelo. Los valores que destacan respectos a los otros obtenidos, son los conseguidos con valores no muy altos en *Max_depth y* valores bajos en *Min_samples_leaf*. En este experimento se lograron valores de *F1-score* en todos los casos mayores que el valor aportado por el modelo de *Regresión Logística*.

Se seleccionaron los valores más altos para si ver si había alguna diferencia más notable en los valores *Precision y Recall* de cada combinación.

- *Max_depth*=17-*Min_samples_leaf* =1→ *F1-score*= 0.705 →*Precision* = 0.671→ *Recall*= 0.742
- *Max_depth*=22-*Min_samples_leaf* =3→ *F1-score*= 0.704→*Precision*= 0.671→ *Recall*= 0.742
- *Max_depth*=17-*Min_samples_leaf* =15→ *F1-score*= 0.704→*Precision*= 0.667→ *Recall*= 0.745

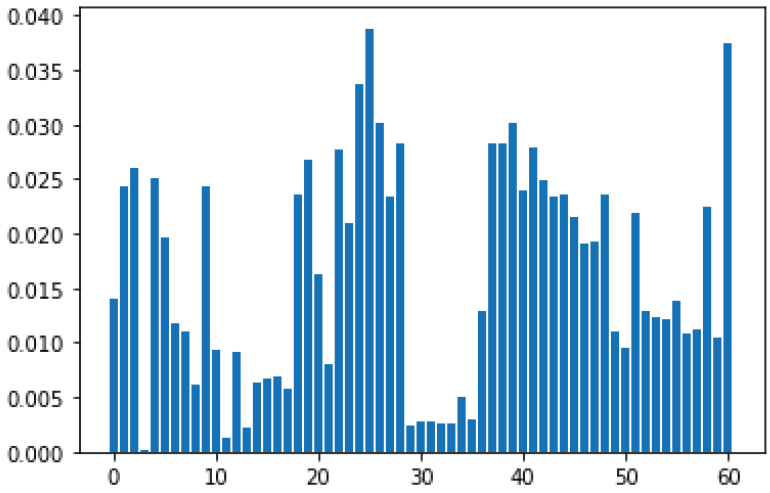

Figura 28. Gráfico donde se representa la importancia de las variables en exp.5

Se puede ver como para estas combinaciones la columna nueva añadida no es la más significativa, pero sí la segunda. La que más influencia obtuvo es *Kw_avg_avg,* no obstante, el valor no supera el 0.04.

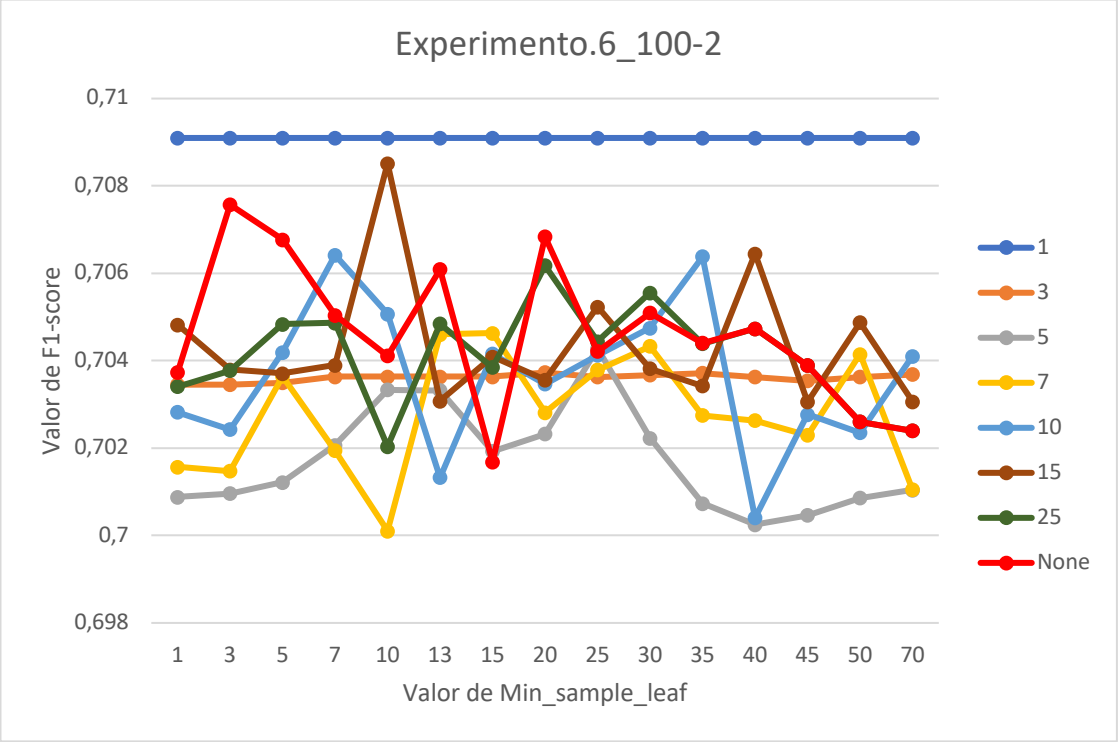

Figura 29. Gráfico donde se muestran los valores de F1-score obtenidos para el exp.6

Los valores de *F1-score* obtenidos para estas diferentes combinaciones están bastante acotados. Se observa que con el valor 1 en el parámetro *Max_depth*, se logró el máximo valor de *F1-score*, además se mantuvo constante, significando que el parámetro *Min_sample_leaf*, no influyó a los modelos al variarlo. Por otro lado, cuando se utilizaron valores como 13 y 15 en el parámetro *Min_sample_leaf*, varias rectas sufren un mínimo, el valor de *F1-score* disminuyó en comparación a los otros valores conseguidos. Se observa como cuando se impuso el valor de 15 en *Max_depth* y un valor de 10 en *Min_sample_leaf*, se produjo un gran aumento del valor *F1-score*, pero no lo suficientemente alto para superar la recta de valor 1 en *Max_depth*.

- *Max_depth*=1-*Min_samples_leaf* =1→ *F1-score*= 0.709→*Precision*= 0.558→ *Recall*= 0.970
- *Max_depth*=1-*Min_samples_leaf* =3→ *F1-score*= 0.709→*Precision*= 0.558→ *Recall*= 0.970
- *Max_depth*=1-*Min_samples_leaf* =5→ *F1-score*= 0.709→*Precision*= 0.558→ *Recall*= 0.970

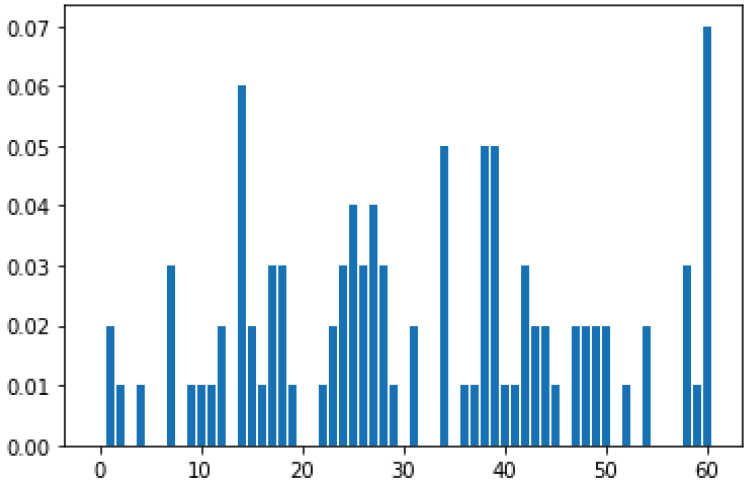

Figura 30. Gráfico donde se representa la importancia de las variables en exp.6

Se puede observar que, para esta combinación, la variable más significativa era la última añadida al modelo. Se diferencian en el gráfico 7 grupos de valores, siendo el más alto superior a 0.07. En los otros experimentos analizados, las variables obtuvieron diferentes valores de importancia. La variable *Data_channel_is_socmed* logra un gran valor a diferencia de otros casos.

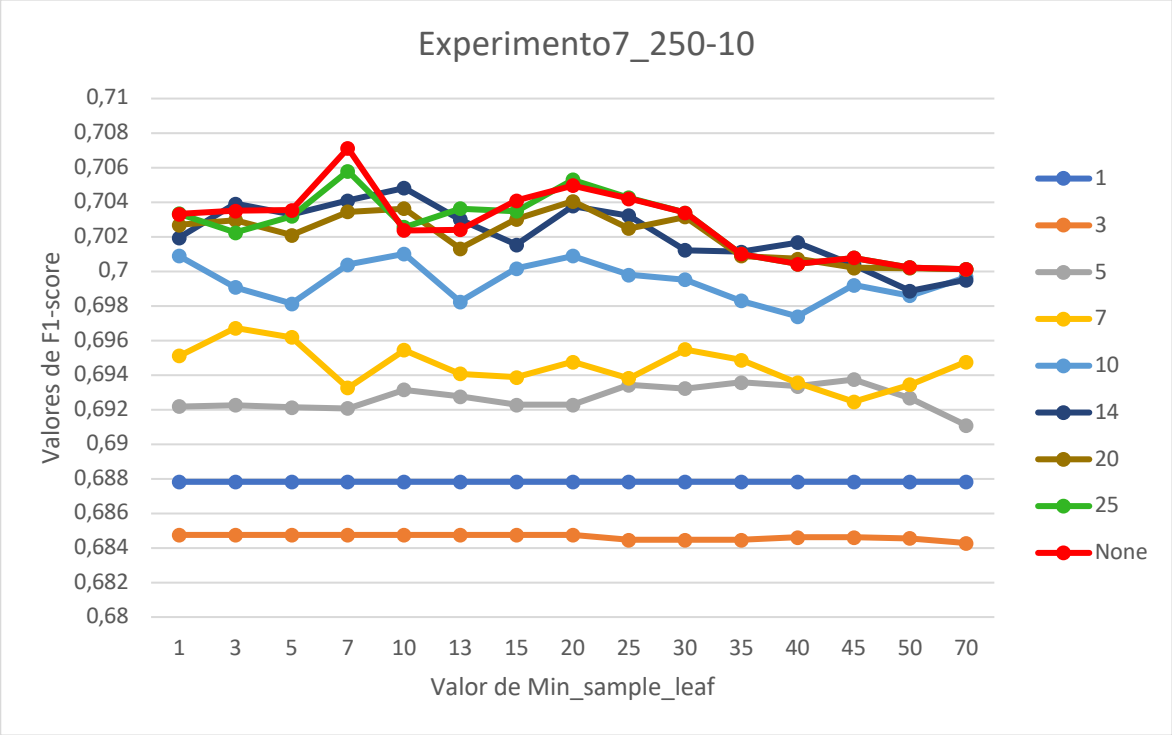

Figura 31. Gráfico donde se muestran los valores de F1-score obtenidos para el exp.7

Se observa una zona la cual está a un rango de valores *F1-score* bastante altos. Esta zona esta compuesta por las rectas formadas de valores altos de los parámetros *Max_depth*, cosa que no se ha visto en experimentos de esta parte. Significa que para un cierto valor de *Max_depth*, los valores obtenidos de *F1-score* no varían mucho, por lo tanto, no influyó lo suficiente modificar este parámetro al modelo. Por otro lado, las rectas obtenidas a partir de valor de *Max_depth* bajos, son los casos en los que se obtuvo menor valor de F1-score.

Respecto el parámetro *Min_sample_leaf*, en los casos que se consiguieron valores bajos de *F1-score*, no se tuvo la suficiente influencia ya que se observan rectas completamente constantes. Con valores logrados altos de *F1-score*, se puede ver que, en la zona descrita anteriormente, este parámetro produjo un pequeño decrecimiento a medida que se iba aumentando.

- *Max_depth*=None-*Min_samples_leaf* =7→ *F1-score*= 0.707→*Precision*= 0.673→ *Recall*= 0.744
- *Max_depth*=35-*Min_samples_leaf* =7→ *F1-score*= 0.707→*Precision*= 0.673→ *Recall*= 0.744
- *Max_depth*=22-*Min_samples_leaf* =3→ *F1-score*= 0.706→*Precision*= 0.670→ *Recall*= 0.745

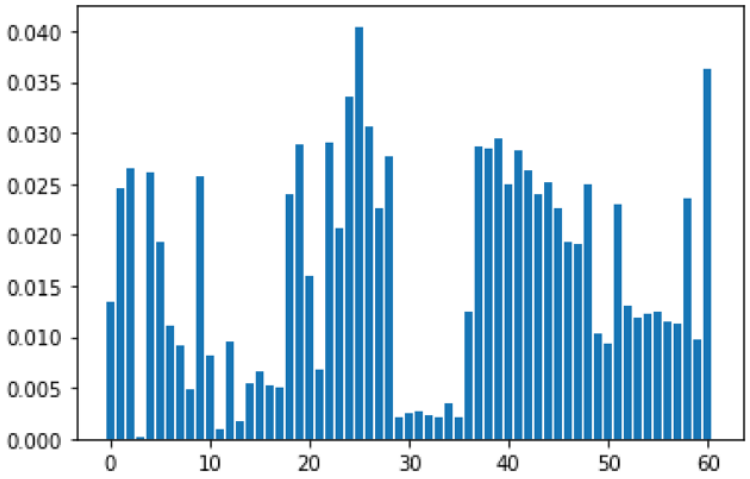

Figura 32. Gráfico donde se representa la importancia de las variables en exp.7

Se aprecia como el máximo valor de importancia en este experimento superó por poco el 0.04. En este caso la variable más importante fue *Kw_avg_avg,* los modelos también tuvieron en consideración la columna añadida pero no de manera tan significativa.

5.7. Comparativa de Resultados

En este apartado se comprarán los cuatro últimos apartados, los apartados de resultados base y nuevas columnas ya se han analizado y comparado.

Durante la realización de los múltiples experimentos, se ha podido observar como los parámetros modificados en ocasiones tenían una gran influencia en las predicciones del modelo, pero en otras dependiendo de las combinaciones algunos parámetros no influían lo suficiente.

Respecto a la primera fase de modificación de parámetros del proyecto, si se hace énfasis en los resultados, se observa como al buscar cuáles eran los parámetros de los árboles de decisión que aportaban una mejor eficiencia al modelo, se seleccionan combinaciones que superan ya el valor de *F1-score* logrado mediante el modelo de *Regresión Logística*. Esto significa que tan solo cambiando dos parámetros el modelo *Random Forest* ya se superaba el valor de *F1-score* ajustándose bien a los datos.

Cuando se realizaron los siete experimentos siguientes, las combinaciones seleccionadas en cada experimento que conseguían el máximo valor de *F1-score*, también eran en las que se lograban valores máximos de *Recall y* en la gran mayoría también *Precision*. La diferencia de valores era bastante pequeña, por lo tanto, si no se dispone de suficiente potencia computacional, sería correcto escoger la combinación en cada experimento que requiriera una potencia menor. No obstante, si se desea la máxima exactitud en las predicciones y se posee bastantes procesadores, entonces sería correcto escoger la mejor opción de cada experimento.

En estos experimentos, en varias ocasiones la columna más significativa era la creada a partir de la combinación de las dos variables más importantes. Al unir estas dos variables aumentó su importancia para el modelo, entonces se puede afirmar que el planteamiento de crear esta nueva columna, mejoraba las predicciones realizadas. Además, las gráficas creadas en cada experimento seguían una misma tendencia, no obstante, los valores de significación de las variables aumentaban o disminuyan.

Comparando los resultados de cada experimento en esta fase, se obtuvo el mejor valor a partir de la combinación que proporcionó el peor valor de *F1-score* seleccionado, valor de *Max_depth* 1 y valor de *Min_sample_leaf* 1, creaba árboles de decisiones muy simples,

solo con una condición ya podía predecir buenos resultados. Esto puede ser debido a que la influencia de los parámetros de *Random Forest* fuese mayor en este experimento que en los otros. En esta combinación el valor *Recall* conseguido es muy superior a los proporcionados por los modelos anteriormente, siendo un valor cercano al 1. Esto significa que el modelo tuvo pocos fallos al predecir una publicación como baja que en realidad era alta, normalmente este valor no superaba el 0.8. En cambio, el valor *Precision* disminuye a diferencia de los logrados en otros experimentos, provocando que el modelo predijera en ocasiones una publicación como alta que en realidad era baja.

Con una profundidad igual a 1 fue en el único experimento donde se logró el máximo valor a partir de utilizar tres variables. En los otros experimentos, se consiguió el máximo valor de *F1-score* y *Recall*, usando en los árboles de decisión dos variables para crearlos. Si se crean árboles de decisión de una profundidad mayor que 1, se pueden seleccionar dos variables aleatoriamente ya que se podrán crear más condiciones. En cambio, si se usan árboles de decisión de profundidad 1, para estos datos es mejor aumentar un poco el número de variables a escoger para que así el modelo pueda hacer una condición más óptima.

Respecto al parámetro *N_estimators*, en bastantes combinaciones se logra el máximo valor de *F1-score* usando un gran número de árboles. En cambio, para profundidad 1, no fue necesario usar tantos como en los otros experimentos. Como conclusión de este suceso, el modelo al tener profundidad 1, si se utilizara más árboles se podría crear un poco de *overfitting* y no generalizar de una forma tan óptima.

En la segunda fase de modificación, sucede igual que en la anterior, en el primer paso todas las combinaciones seleccionadas superaron el valor obtenido de *F1-score* del modelo de *Regresión Logística*. Modificando dos parámetros del *Random Forest* las predicciones ya eran buenas.

Los valores más altos de *F1-score* se consiguieron, en la gran mayoría de las combinaciones seleccionadas, utilizando solo dos variables aleatorias para crear los árboles de decisión. Respecto el número de árboles ha utilizar, el mejor caso que se seleccionó no era la combinación donde se usaba el número máximo. En este primer paso, a diferencia de la primera fase de modificación, se obtuvieron valores más altos de *F1-*

score, esto quiere decir que, los parámetros de *Random Forest* tienen más influencia en el modelo que los parámetros de los árboles de decisión.

Cuando se realizaron los siete experimentos de esta fase, las combinaciones de cada experimento donde se obtuvo el máximo valor de *F1-score* siempre era con valores del parámetro *Min_sample_leaf* bajos. En este caso no se necesitaba una gran cantidad de datos para crear un nodo, esto puede ser ocasionado por no utilizar muchas filas en los árboles.

En estos experimentos la diferencia de valores no era muy significativa, en cada experimento se diferenciaban por solo 0.001 décimas. No obstante, se buscó siempre la mejor situación.

Respecto a los parámetros de los árboles de decisión, en esta fase a diferencia de la anterior, se obtuvo el máximo valor de *F1-score* a partir de árboles sin límite de profundidad. Este valor fue superior que el máximo logrado en la anterior fase del proyecto, además era muy superior al valor proporcionado por el modelo de *Regresión Logística*.

Las combinaciones con las que se obtuvo el máximo valor de *F1-score* en cada experimento no se lograba el máximo valor de *Recall*, en cambio el valor de *Precision* si que era el máximo. En la única situación en la que se logró a la misma vez el máximo valor de *F1-score* y *Recall*, fue el caso donde la profundidad límite era 1 pero el valor *Precision* disminuyó.

Para realizar una buena comparación de los mejores valores de *F1-score* logrados en cada fase se muestra a continuación las dos matrices de confusión y las métricas obtenidas.

	0	1
0	1205	4342
1	468	5879

Tabla 8. Tabla con las predicciones realizadas en la primera fase para combinación de mejor F1-score.

	0	1
0	3120	2427
1	1505	4842

Tabla 9. Tabla con las predicciones realizadas en la segunda fase para combinación de mejor F1-score.

- Primera Fase → Max_depth=1, Min_sample=1, Maxfeat=2, N_estim=150→ *F1-score*= 0.710→*Precision*=0.570→*Recall*=0.942

- Segunda Fase→ Max_depth=None, Min_sample=5, Maxfeat=2, N_estim=250→ *F1-score*=0.711→*Precision*=0.666→*Recall*=0.762

Se observa como los valores de *F1-score* son muy similares, la diferencia es de 0.001. En la primera fase el valor logrado de la métrica *Recall* es mayor que en de la segunda fase. En cambio, el valor de *Precision* es superior en la segunda fase que en la primera.

Respecto a estos dos casos, es necesaria una potencia computacional mayor en la combinación lograda en la segunda fase. Se crean un número mayor de árboles y, además, no hay límite de profundidad en los árboles de decisión.

La elección de cuál de los casos es mejor depende del criterio que tenga la empresa. Si se prefiere un modelo que realice predicciones las cuales tengan una buena tasa de acierto al predecir una publicación como alta y, además, no tener un gran número de fallos al predecir una publicación como baja que en realidad es alta, el modelo a escoger sería la combinación obtenida en la segunda parte.

No obstante, si la empresa se quiere centrar en que el modelo tenga muy pocos fallos a la hora de decidir que una noticia es de baja popularidad y en realidad es de alta, la combinación obtenida en la primera fase sería la idónea.

5.8. Experimento adicional

El objetivo de este experimento es poder observar como predice el modelo de *Random Forest*, dejando los todos los parámetros por defecto, a partir de los datos con tres categorías en la columna de *labels*. En este caso se quiso ajustas las categorías de baja y alta, por eso hay más datos categorizados como popularidad media. En estos datos igual que en los datos con dos categorías se usó la validación *holdout*, distribuyendo un 30% como datos de prueba y 70% como datos de entrenamiento. Se realizó una buena distribución de las diferentes categorías para cada tipo de datos, para evitar *overfitting*.

	precision	recall	f1-score	support
alt	0.39	0.35	0.37	3127
baix	0.43	0.36	0.39	3006
mig	0.52	0.60	0.55	5761
accuracy			0.47	11894
macro avg	0.45	0.43	0.44	11894
weighted avg	0.46	0.47	0.46	11894

Tabla 10. Métricas de evaluación para datos con tres categorías con RF

Se puede observar como las publicaciones con una popularidad altas es donde se obtuvo un peor valor de *F1-score*. Tanto en la categoría de baja como en la de alta, el valor de *F1-score* es bastante bajo, siendo inferior a 0.5, significa que el modelo para estos dos tipos no se ajustó adecuadamente y no generalizaba de forma óptima.

En comparación, el valor de *F1-score* logrado para predicciones de la categoría de media, fue superior a los otros dos valores de *F1-score,* aunque no superaba a los valores logrados con los datos de dos categorías ni logrado con *Regresión Logística*.

Al haber tres categorías, pudo provocar que el modelo le costase realizar buenas predicciones. Modificando los parámetros del modelo tanto los de *Random Forest* como los de los árboles de decisión, el modelo puede que obtuviera mejores predicciones y generalizara de una forma mucho mejor.

Las métricas de evaluación se obtuvieron a partir de los resultados que se muestran en la matriz siguiente:

	0	1	2
0	1098	366	1663
1	414	1076	1516
2	1274	1059	3428

Tabla 11. Tabla con las predicciones realizadas para datos con tres categorías con RF

Por añadir, para empresa una noticia con una popularidad media puede que no le aporte mucha información, ya que las empresas suelen preferir identificar las noticias con popularidad baja para rectificarlas y las noticias de popularidad alta para saber cuales son con las que obtienen un beneficio mayor. Se podría haber distribuido los dados de manera que todos tuvieran el mismo número de muestras en cada categoría, pero el modelo no habría estado tan ajustado y habría márgenes más pequeños.

6. Impacto ambiental

Este proyecto no comporta gran influencia ambiental, ya que no se crea ningún producto físico, sino que todo se lleva a cabo digitalmente.

La herramienta con la que se ha realizado el proyecto ha sido con un ordenador. El impacto que este puede causar ambientalmente es el consumo eléctrico que conlleva cargarlo para poder utilizarlo, sino sería inservible, y los materiales que se han usado para crearlo.

El consumo eléctrico es bastante pequeño, ya que los ordenadores de nueva generación tienen una mayor duración de batería y no es necesario cargarlos todo el rato. Por otra parte, los ordenadores son herramientas fundamentales en nuestra vida hoy en día, actualmente pocas personas no tienen un ordenador en casa ya que se usan para una gran cantidad de finalidades. En este caso se usó el ordenador destinado para trabajos y apuntes de la universidad y no hizo falta comprar uno nuevo

7. Planificación

A continuación, se muestra la planificación que se ha seguido para realizar el trabajo durante el tiempo disponible.

Fases del proyecto	Acciones realizadas	ago-19		sept-19				oct-19				nov-19				dic-19				ene-20		
		34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	1	2	3
Fase Inical	Búsqueda de información sobre minería de datos																					
	Estudiar metodología CRISP-DM																					
	Obtener conocimientos en Pandas y Sklearn																					
Comprensión y preparación de datos	Comprensión de las variables																					
	Preparación de datos																					
	Transformación de datos																					
Modelaje y Evaluación	Estudiar los modelos de predicción y tipos de validación																					
	Aprender las diferentes métricas de evaluación																					
	Construcción y realización de experimentos																					
	Analizar y comparar los experimentos																					
Fase final	Redactar la memoria del proyecto																					

Tabla 12. Tabla donde se muestra la planificación del proyecto

8. Presupuesto

A continuación, se detalla en una tabla el tiempo dedicado y el precio por hora en cada fase realizada en el proyecto.

Fases	**Tiempo de dedicación**	**Precio por hora**	**Coste total en cada fase**
Búsqueda de información	130h	25€	3250€
Comprensión y preparación de datos	40h	30€	1200€
Modelaje y evaluación	100h	40€	4000€
Redacción de memoria	70h	20€	1400€
Total	340h		9850€

Tabla 13. Tabla donde muestra los costes del proyecto no totales

Se decidió distribuir el presupuesto del proyecto en cuatro fases importantes, las cuales pertenecen a la metodología CRISP-DM. La fase de búsqueda de información abarca todo el tiempo dedicado a investigar sobre los temas de los que no se poseía el suficiente conocimiento.

En la redacción de memoria se ha contabilizado un total de 70 horas. Esto es debido ha que se ha explicado todo lo que se ha realizado en el proyecto y lo que se ha tenido presente en cada caso. Además, se ha redactado todos los conceptos importantes aprendidos durante la realización del proyecto.

El precio por hora en cada fase va variando ya que es más costoso realizar el modelaje y la evaluación de los modelos en comparación con la redacción de la memoria. Se dedicó una gran cantidad de horas para la búsqueda de información, es una fase importante para el proyecto ya que no se ha realizado ninguna asignatura la cual detalle específicamente minería de datos. Además, se debía de tener clara la metodología que se quería seguir.

Respecto a las herramientas empleadas en el proyecto, no se ha tenido la obligación de comprar un ordenador nuevo ya que se ha usado el que se había adquirido destinado para la universidad. Se debe tener presente la amortización al haber empleado un total de 340 horas para realizar el trabajo únicamente haciendo servir el ordenador. Los programas utilizados en el proyecto son gratuitos, por lo tanto, no se aplica ningún coste por las licencias.

El ordenador costó 1500€ al ser un ordenador potente para soportar programas de diseño, según la agencia tributaria un ordenador tiene un 25% de amortización anual para un uso de 1200 horas al año. Realizando los cálculos se obtiene que la amortización es de 375€ durante un año. En este caso solo se utiliza un 28,3% de horas respecto al total de horas en un año, así que usar el ordenador conlleva un coste de 106,25€. Sumando los dos tipos de costes el presupuesto del estudio es un total de 9956.25€.

9. Trabajos futuros

El estudio de estos datos se podría haber enfocados desde diferentes puntos de vista, dando lugar a nuevos trabajos o ampliaciones del realizado.

Otro posible trabajo a realizar a posteriori sería, en vez de poner énfasis en mejorar los resultados predichos para la categoría de alta popularidad, enfocar el proyecto en mejorar las predicciones para la categoría de baja popularidad. Para una empresa puede que no sea tan beneficioso, pero se podrían dar casos en que se obtendrían diferentes parámetros a los logrados, de los modelos utilizados y así compararlos con los conseguidos en este estudio.

Por otro lado, en este estudio se ha usado todas las columnas de los datos para construir los diferentes modelos de predicción. Se pudo observar cómo si se añadían más variables, los modelos mejoraban, pero no si se suprimían variables. Cuando se creó un gráfico para poder saber cómo de significativa era cada variable, se observó que había variables las cuales no aportaban información al modelo. De este modo, se puede hacer un trabajo con los mismos datos, pero eliminando las columnas que no son significativas a la hora de realizar predicciones para los modelos. Usando los mismos modelos se podría ver cómo mejoran o empeoran las predicciones realizadas.

En este proyecto, se decidió modificar los parámetros de *Random Forest* únicamente, ya que se consideraron más relevantes que los de *Regresión Logística*. El modelo de *Regresión Logística* se creó usando los parámetros por defecto que el modelo tiene. Se podría ampliar el conocimiento de los parámetros que contiene y modificarlos en un trabajo, para observar si a partir de estos parámetros el modelo realiza mejores predicciones y supera los valores de *F1-score* , *Recall* y *Precision* logrados por medio de *Random Forest*. Además, se podría ampliar los parámetros a modificar en *Random Forest*. Al disponer de poco tiempo se consideraros importantes los descritos anteriormente.

Se podría enfocar un nuevo trabajo usando otros modelos de predicción para los mismos datos, en este estudio se decidió solo usar dos modelos ya que no se disponía de más tiempo. *Random Forest* y *Regresión Logística* son unos de los más utilizados en minería de datos y su comprensión es rápida, pero existen otros tipos de modelos como *Kneighbors* o *SVM*, los cuales no se han aplicado en este proyecto. Construir diferentes modelos predictores para unos mismos datos puede ser beneficioso para las empresas para tener diferentes puntos de vista y realizar más comparaciones con el fin de conseguir que el modelo se ajuste mejor a los datos.

En este proyecto se decidió utilizar los datos de dos categorías. No obstante, se realizó un experimento donde se creaba un modelo de *Random Forest*, con todos los parámetros por defecto, a partir de datos con tres categorías. Se podría fijar el objetivo de usar los datos de tres categorías para ver si se logra obtener mejores predicciones de cada categoría de popularidad mediante los mismos modelos. Se pueden dar casos donde modificando otros parámetros de los modelos utilizados, se obtengan mejores valores de *F1-score*, *Recall o Precision* y así poder ver con mayor exactitud cómo de populares serán las publicaciones con más variedad de popularidad. Otro proyecto es usar otros modelos los cuales se ajusten mejor a datos con *labels* con más de dos tipos.

Conclusiones

En el presente trabajo se han conseguido predecir la popularidad de las noticias de la pagina web *Mashable*, siendo el objetivo principal del proyecto. Este objetivo se ha logrado con una buena aplicación de la metodología CRSIP-DM. Mediante esta metodología se ha podido realizar un buen estudio de los datos proporcionados y crear modelos con los que se conseguían buenas predicciones. Se tuvo que volver atrás en una de las fases para añadir nuevas columnas, pero al ser una metodología cíclica no hubo inconvenientes. Así pues, se ha podido razonar todas las decisiones que se han ido tomando durante todo el proyecto.

Se ha podido hacer un buen estudio sobre el modelo *Random Forest*, y cuando se puso en practica se observó la importancia de los parámetros escogidos para modificar. Normalmente estos parámetros mejoraban las predicciones sobre las publicaciones, aunque en otros casos provocaban que el modelo bajara la tasa de acierto. Algunos parámetros estaban notablemente restringidos ya que, solo se realizaron mediante un ordenador, de manera que, por ejemplo, el número de arboles a crear se limitaba mucho. Normalmente cuando se hacen proyectos así las empresas proporcionan procesadores de gran capacidad, los cuales permiten formar una gran cantidad de modelos en poco tiempo. Ocurre lo mismo con el uso de la función *GridSearch*, con la que el proyecto habría obtenido mejores resultados ya que la función habría hecho una comparación mucho más compleja.

Respecto a la *Regresión Logística*, se ha podido comprobar que es un tipo de modelo inferior a *Random Forest*. Sin embargo, por falta de tiempo no se pudieron modificar los parámetros, pero aún así se pudo comprobar que de esta manera mediante *la Regresión Logística* se lograban buenas predicciones y el modelo generalizaba de una forma eficiente.

Personalmente, haber realizado el trabajo con el lenguaje de programación Python, ha sido muy beneficioso ya que he aprendido nuevos conocimientos. Un ejemplo es saber usar las librerías Pandas. Además, el tiempo de programación se ha disminuido, antes para a mi era más costoso programar y conllevaba a una mayor dedicación.

Hacer este trabajo me ha servido para valorar más la importancia del tiempo a la hora de realizar un estudio de esta magnitud. Una buena distribución del tiempo para el trabajo es fundamental, ya que pueden surgir imprevistos de último momento los cuales provoquen malas decisiones y limitar o no hacer todo lo que se deseaba en el proyecto.

La minería de datos es fundamental para la sociedad en la que se vive hoy en día, sobre este tema en el grado se debería de hablar más y estudiarlo más a fondo. Se puede logran grandes beneficios y objetivos, si se aplica bien.

Como conclusión personal, al haber realizado este proyecto con un tema el cual nunca había estudiado, me ha sido bastante costoso ya que debía mirar mucha información y entender conceptos bastante complejos. Valoro positivamente este proyecto, he aprendido conocimientos nuevos sobre este sector de la ingeniería, convirtiéndome en un mejor profesional. A la hora de buscar empleo, muchas empresas empiezan a tener como requisito saber minería de datos, mi perfil se podrá ajustará mejor a las vacantes disponibles y dispondré de mayores oportunidades de poder lograr un buen trabajo.

Agradecimientos

Quiero agradecer a mi tutor Lluis Talavera el tiempo que ha dedicado a explicarme todos los conceptos que no llegaba a entender bien.

Un agradecimiento especial a toda mi familia y mi pareja por estar a mi lado durante todo este tiempo.

Bibliografia

[1] DAVID L. OLSON, DURSUN DELEN. *Advanced Data Mining Techniques*, 2008, 169 p ISBN: 978-3-540-76916-3

[2] Varios autores. *Anaconda Distribution* [https://www.anaconda.com/distribution/]

[3] Varios autores. *Python Data Analysis Library. Pandas 0.25.3 documentation* [https://pandas.pydata.org/pandas-docs/stable/user_guide/index.html]

[4] Varios autores. *Towads data science Sharing concepts, ideas and codes* [https://towardsdatascience.com/]

[5] *Scikit-learn Machine Learning in Pyhton* [https://scikit-learn.org/stable/index.html#]

[6] *Aprende Machine learning* [https://www.aprendemachinelearning.com/]

[7] Varios autores. *Stack overflow questions* [https://stackoverflow.com/questions/]

[8] Cashmore, P. *Mashable Benelux* [https://mashable.com/?europe=true]

[9] Cristina Gil. *GItHub* [https://github.com/CristinaGil/Ciencia-de-Datos-R]

[10] Villa, M. *Pyciencia* [http://pyciencia.blogspot.com/]

[11] *UCI Machine learning Repository* (Base de datos pública) [https://archive.ics.uci.edu/ml/index.php]

[12] Varios autores. *Learn OpenCV* [https://www.learnopencv.com/]

[13] *Agencia Tributaria* [https://www.agenciatributaria.es/]

Anexos

A1.Experimentos

1	3	5	7	10	13	14	15	17	20	22	25	30	35	None	MAXDEP
0.69376	0.6843	0.6918	0.6927	0.6909	0.6819	0.68	0.6813	0.6696	0.6652	0.6589	0.6439	0.6308	0.6345	0.6358	**1**
0.69376	0.6843	0.6921	0.6944	0.6934	0.6847	0.684	0.6785	0.6741	0.6705	0.6709	0.6686	0.6679	0.6664	0.6643	**3**
0.69376	0.6843	0.6922	0.6962	0.6916	0.6845	0.6836	0.6841	0.6782	0.6758	0.6757	0.6795	0.6727	0.6714	0.6714	**5**
0.69376	0.6843	0.6919	0.6936	0.6915	0.6888	0.6952	0.6825	0.683	0.679	0.6743	0.6795	0.6768	0.6826	0.6826	**7**
0.69376	0.6843	0.6911	0.6978	0.6942	0.689	0.6888	0.6868	0.6838	0.6825	0.6848	0.6851	0.6817	0.6817	0.6817	**10**
0.69376	0.6843	0.6899	0.6979	0.6909	0.6914	0.6899	0.6873	0.6904	0.6848	0.6857	0.6854	0.6854	0.6854	0.6854	**13**
0.69376	0.6843	0.6902	0.6983	0.6915	0.6925	0.6884	0.6875	0.6903	0.6895	0.6876	0.6908	0.6882	0.6882	0.6882	**15**
0.69376	0.6843	0.6905	0.6986	0.6935	0.6963	0.6956	0.6891	0.6898	0.6863	0.6859	0.6855	0.6855	0.6855	0.6855	**20**
0.69376	0.6843	0.6905	0.693	0.6963	0.6955	0.6919	0.6959	0.6935	0.6903	0.693	0.6925	0.6925	0.6925	0.6925	**25**
0.69376	0.6843	0.6895	0.6989	0.6953	0.6997	0.6935	0.6924	0.6925	0.6943	0.6937	0.6937	0.6937	0.6937	0.6937	**30**
0.69376	0.6843	0.6895	0.6948	0.6917	0.6953	0.6941	0.6911	0.6938	0.6943	0.6924	0.6924	0.6924	0.6924	0.6924	**35**
0.69376	0.6842	0.6918	0.6924	0.6962	0.6894	0.6935	0.6936	0.695	0.6955	0.6955	0.6955	0.6955	0.6955	0.6955	**40**
0.69376	0.6842	0.6905	0.6931	0.696	0.7008	0.6904	0.6922	0.6928	0.692	0.692	0.692	0.692	0.692	0.692	**45**
0.69376	0.6842	0.6892	0.6937	0.6942	0.6943	0.6951	0.6966	0.6952	0.6947	0.6949	0.6949	0.6949	0.6949	0.6949	**50**
0.69376	0.6842	0.6927	0.6958	0.6936	0.6948	0.695	0.6929	0.6925	0.6925	0.6925	0.6925	0.6925	0.6925	0.6925	**70**
															MINSAMP

Tabla 14. Tabla donde se muestra los valores de F1-score para los parámetros del Árbol de decisión primera fase

2	3	5	7	10	20	30	40	auto	log2	None	Maxfeatures
0.659618	0.669714	0.668225	0.668461	0.66853	0.672164	0.672484	0.669548	0.668461	0.668225	0.670355	**2**
0.685436	0.687959	0.688236	0.688056	0.684104	0.687907	0.685134	0.683306	0.688056	0.688236	0.678916	**5**
0.694926	0.695595	0.695626	0.700758	0.69295	0.694238	0.696194	0.695161	0.700758	0.695626	0.685506	**10**
0.69706	0.700672	0.695981	0.699734	0.696195	0.69469	0.696601	0.693696	0.699734	0.695981	0.690206	**15**
0.699701	0.700401	0.699312	0.700392	0.701247	0.695535	0.696569	0.694313	0.700392	0.699312	0.69366	**25**
0.702224	0.701695	0.701104	0.700118	0.701747	0.697581	0.698247	0.697133	0.700118	0.701104	0.696029	**50**
0.701504	0.701708	0.702683	0.703026	0.70188	0.699948	0.699597	0.696622	0.703026	0.702683	0.694918	**75**
0.700516	0.700044	0.703135	0.702779	0.701804	0.699657	0.69941	0.696707	0.702779	0.703135	0.696705	**85**
0.703083	0.701249	0.70157	0.701345	0.701032	0.699143	0.699366	0.696999	0.701345	0.70157	0.696237	**100**
0.704382	0.702446	0.702112	0.701773	0.700616	0.697914	0.700291	0.696913	0.701773	0.702112	0.695918	**125**
0.705823	0.701506	0.704005	0.701625	0.701582	0.697408	0.699255	0.696861	0.701625	0.704005	0.695808	**150**
0.70578	0.702162	0.701853	0.70122	0.701314	0.698973	0.699821	0.698804	0.70122	0.701853	0.696295	**200**
0.706851	0.702478	0.702176	0.701612	0.700594	0.700171	0.69851	0.697713	0.701612	0.702176	0.696587	**250**
											Nestimators

Tabla 15. Tabla donde se muestran los valores de F1-score en exp.1 en primera fase

2	3	5	7	10	20	30	40	auto	log2	None	Maxfeatures
0.671678	0.661775	0.663034	0.676952	0.666871	0.679069	0.670059	0.680299	0.676952	0.663034	0.66754	**2**
0.686778	0.682833	0.686094	0.693119	0.679101	0.68808	0.685659	0.688495	0.693119	0.686094	0.67113	**5**
0.699562	0.69383	0.697334	0.698922	0.689058	0.689578	0.689815	0.686657	0.698922	0.697334	0.675299	**10**
0.702775	0.698755	0.696342	0.700584	0.6931	0.689614	0.692703	0.686587	0.700584	0.696342	0.679623	**15**
0.701629	0.698447	0.695557	0.698355	0.69195	0.690473	0.690263	0.688301	0.698355	0.695557	0.681873	**25**
0.701892	0.700288	0.69709	0.699207	0.693063	0.689732	0.690535	0.688061	0.699207	0.69709	0.685418	**50**
0.703248	0.699417	0.697244	0.69848	0.694198	0.693437	0.69229	0.688309	0.69848	0.697244	0.685959	**75**
0.702437	0.700454	0.697435	0.69909	0.692843	0.692468	0.691121	0.687992	0.69909	0.697435	0.685168	**85**
0.704334	0.700756	0.697746	0.69777	0.693304	0.692336	0.691636	0.688252	0.69777	0.697746	0.684408	**100**
0.703345	0.699863	0.69871	0.697415	0.692598	0.691786	0.690803	0.690254	0.697415	0.69871	0.685564	**125**
0.703962	0.702314	0.699225	0.697415	0.694481	0.69351	0.690963	0.689402	0.697415	0.699225	0.684876	**150**
0.703638	0.701532	0.698874	0.698471	0.69562	0.694209	0.691727	0.69017	0.698471	0.698874	0.684484	**200**
0.70472	0.701245	0.698517	0.697774	0.695485	0.694546	0.691756	0.689779	0.697774	0.698517	0.685487	**250**
											Nestimators

Tabla 16. Tabla donde se muestran los valores de F1-score en exp.2 en primera fase

2	3	5	7	10	20	30	40	auto	log2	None	Maxfeat
0.667468	0.674134	0.664753	0.663902	0.670844	0.670357	0.671959	0.670057	0.663902	0.664753	0.666109	**2**
0.684001	0.687863	0.686194	0.684851	0.686826	0.685194	0.680048	0.683678	0.684851	0.686194	0.678261	**5**
0.69265	0.696439	0.693841	0.696574	0.694977	0.692832	0.69144	0.69046	0.696574	0.693841	0.686649	**10**
0.699331	0.700747	0.69625	0.698944	0.696791	0.695484	0.695835	0.694588	0.698944	0.69625	0.691412	**15**
0.700994	0.69967	0.697564	0.700967	0.698474	0.699836	0.698351	0.693553	0.700967	0.697564	0.695495	**25**
0.703166	0.699164	0.701089	0.699926	0.699405	0.700216	0.700862	0.696049	0.699926	0.701089	0.695287	**50**
0.704348	0.698804	0.701222	0.699859	0.701778	0.699649	0.699739	0.695711	0.699859	0.701222	0.69614	**75**
0.703303	0.699743	0.700556	0.700974	0.7009	0.700373	0.700359	0.695958	0.700974	0.700556	0.696732	**85**
0.704875	0.700176	0.701406	0.701075	0.700677	0.699478	0.698887	0.696478	0.701075	0.701406	0.696445	**100**
0.705908	0.700621	0.700628	0.700296	0.699546	0.699433	0.699821	0.696872	0.700296	0.700628	0.695775	**125**
0.706138	0.701739	0.70068	0.702115	0.700475	0.700857	0.699223	0.697355	0.702115	0.70068	0.695899	**150**
0.707271	0.700629	0.701832	0.701967	0.701512	0.70143	0.699911	0.697598	0.701967	0.701832	0.696399	**200**
0.707388	0.701498	0.703723	0.700177	0.701957	0.701423	0.698986	0.697647	0.700177	0.703723	0.695009	**250**
											Nestimators

Tabla 17. Tabla donde se muestran los valores de F1-score en exp.3 en primera fase

2	3	5	7	10	20	30	40	auto	log2	None	Maxfeat
0.650431	0.651166	0.663991	0.669031	0.659954	0.662251	0.658338	0.661531	0.669031	0.663991	0.652531	**2**
0.680116	0.683429	0.683849	0.681699	0.677712	0.685138	0.677578	0.684023	0.681699	0.683849	0.669051	**5**
0.698935	0.694048	0.691196	0.6956	0.691486	0.690523	0.68992	0.689567	0.6956	0.691196	0.682694	**10**
0.701186	0.698961	0.697073	0.693569	0.694943	0.696402	0.693177	0.693669	0.693569	0.697073	0.690206	**15**
0.703138	0.701048	0.698214	0.699888	0.69748	0.698211	0.693323	0.696226	0.699888	0.698214	0.689837	**25**
0.705479	0.703145	0.702254	0.700433	0.70161	0.700644	0.697504	0.698589	0.700433	0.702254	0.695404	**50**
0.706337	0.701105	0.702771	0.70035	0.701773	0.701697	0.697263	0.699228	0.70035	0.702771	0.69477	**75**
0.706457	0.703117	0.702575	0.70067	0.702594	0.702077	0.697256	0.699108	0.70067	0.702575	0.694718	**85**
0.707349	0.705242	0.70255	0.702566	0.700923	0.700515	0.697949	0.701176	0.702566	0.70255	0.695593	**100**
0.706919	0.704983	0.702827	0.69997	0.702308	0.701875	0.697761	0.700935	0.69997	0.702827	0.696277	**125**
0.707766	0.703987	0.702962	0.701189	0.703334	0.70124	0.698945	0.701137	0.701189	0.702962	0.696995	**150**
0.708006	0.705588	0.702435	0.703073	0.702288	0.702388	0.700321	0.701376	0.703073	0.702435	0.69781	**200**
0.708898	0.706055	0.704073	0.703249	0.703792	0.701374	0.701951	0.70065	0.703249	0.704073	0.698839	**250**
											Nestimators

Tabla 18. Tabla donde se muestran los valores de F1-score en exp.4 en primera fase

2	3	5	7	10	20	30	40	auto	log2	None	Maxfeatures
0.662553	0.667419	0.659346	0.668233	0.667273	0.666057	0.668173	0.661301	0.668233	0.659346	0.662561	**2**
0.684346	0.684995	0.680842	0.688929	0.685723	0.686882	0.679885	0.683519	0.688929	0.680842	0.672863	**5**
0.696585	0.694835	0.690457	0.695516	0.693102	0.696622	0.69022	0.687098	0.695516	0.690457	0.683012	**10**
0.699256	0.698394	0.694219	0.697144	0.698192	0.697629	0.694043	0.691556	0.697144	0.694219	0.68725	**15**
0.702038	0.701444	0.698534	0.698274	0.699859	0.697626	0.694735	0.692198	0.698274	0.698534	0.690485	**25**
0.703984	0.703255	0.701162	0.69997	0.699732	0.700531	0.694266	0.693577	0.69997	0.701162	0.696075	**50**
0.704744	0.701903	0.701783	0.700037	0.699866	0.701322	0.695958	0.695933	0.700037	0.701783	0.696296	**75**
0.703889	0.703858	0.702431	0.69884	0.700876	0.699813	0.696068	0.695913	0.69884	0.702431	0.694956	**85**
0.705771	0.704083	0.702304	0.69925	0.70003	0.698826	0.698223	0.696109	0.69925	0.702304	0.69562	**100**
0.706395	0.703853	0.701785	0.700617	0.700171	0.700418	0.697855	0.69788	0.700617	0.701785	0.695711	**125**
0.706199	0.703332	0.701884	0.702365	0.701144	0.70065	0.699051	0.698158	0.702365	0.701884	0.696791	**150**
0.706566	0.703763	0.700362	0.701324	0.700706	0.700926	0.699067	0.699298	0.701324	0.700362	0.696577	**200**
0.70607	0.703652	0.70236	0.699666	0.700721	0.700679	0.697984	0.69875	0.699666	0.70236	0.697713	**250**
											Nestimators

Tabla 19. Tabla donde se muestran los valores de F1-score en exp.5 en primera fase

2	3	5	7	10	20	30	40	auto	log2	None	Maxfeatures
0.658696	0.675486	0.665519	0.672072	0.664747	0.667877	0.667425	0.669808	0.672072	0.665519	0.666413	**2**
0.688828	0.691795	0.685655	0.690129	0.683095	0.683645	0.682835	0.681663	0.690129	0.685655	0.678607	**5**
0.698263	0.698616	0.693596	0.695237	0.689702	0.694299	0.691597	0.689337	0.695237	0.693596	0.686558	**10**
0.702108	0.702355	0.695755	0.701353	0.692577	0.696555	0.69748	0.694908	0.701353	0.695755	0.692308	**15**
0.701673	0.700345	0.698495	0.701126	0.694563	0.693994	0.696945	0.695059	0.701126	0.698495	0.695352	**25**
0.704071	0.700022	0.701679	0.701374	0.695646	0.696515	0.699566	0.696754	0.701374	0.701679	0.69519	**50**
0.704869	0.698991	0.703214	0.700573	0.698585	0.695691	0.699342	0.697835	0.700573	0.703214	0.696147	**75**
0.703942	0.700526	0.703518	0.701731	0.698384	0.696108	0.697042	0.698165	0.701731	0.703518	0.696933	**85**
0.703559	0.700584	0.702787	0.702069	0.698564	0.696875	0.698844	0.697448	0.702069	0.702787	0.696607	**100**
0.707428	0.701087	0.702447	0.70235	0.69813	0.697727	0.698754	0.697344	0.70235	0.702447	0.695542	**125**
0.706435	0.701183	0.703561	0.703199	0.699035	0.699067	0.699888	0.69601	0.703199	0.703561	0.696198	**150**
0.705721	0.701882	0.701425	0.703074	0.699488	0.699016	0.699605	0.697267	0.703074	0.701425	0.696295	**200**
0.706511	0.703085	0.702555	0.703526	0.700987	0.699172	0.69847	0.697754	0.703526	0.702555	0.695399	**250**
											Nestimators

Tabla 20. Tabla donde se muestran los valores de F1-score en exp.6 en primera fase

2	3	5	7	10	20	30	40	auto	log2	None	Maxfeatures
0.695905	0.695905	0.695905	0.695905	0.681001	0.632311	0.642601	0.642601	0.695905	0.695905	0.603599	**2**
0.69778	0.697487	0.693483	0.693483	0.700581	0.662649	0.642601	0.642601	0.693483	0.693483	0.599476	**5**
0.69974	0.699227	0.697013	0.693755	0.693051	0.634284	0.617176	0.602766	0.693755	0.697013	0.599476	**10**
0.706127	0.706566	0.700475	0.696932	0.687838	0.644098	0.639205	0.609799	0.696932	0.700475	0.599656	**15**
0.708716	0.702195	0.697499	0.688173	0.685315	0.643135	0.639524	0.613114	0.688173	0.697499	0.598246	**25**
0.709304	0.703828	0.699385	0.689114	0.681046	0.6517	0.639825	0.635344	0.689114	0.699385	0.599656	**50**
0.708439	0.708128	0.70312	0.693449	0.671469	0.645877	0.640126	0.631474	0.693449	0.70312	0.599656	**75**
0.709512	0.707911	0.701493	0.690245	0.66633	0.644858	0.639512	0.618649	0.690245	0.701493	0.599656	**85**
0.709098	0.709681	0.70592	0.695369	0.674364	0.65168	0.640204	0.621982	0.695369	0.70592	0.599656	**100**
0.708703	0.710793	0.706552	0.698164	0.677372	0.644063	0.639349	0.62149	0.698164	0.706552	0.599656	**125**
0.707833	0.710839	0.706381	0.69706	0.675039	0.644493	0.640253	0.621933	0.69706	0.706381	0.599656	**150**
0.707912	0.709888	0.706628	0.699791	0.681664	0.652776	0.639355	0.622994	0.699791	0.706628	0.599706	**200**
0.705809	0.709988	0.709876	0.700848	0.687824	0.653094	0.640607	0.62417	0.700848	0.709876	0.60111	**250**
											Nestimators

Tabla 21. Tabla donde se muestran los valores de F1-score en exp.7 en primera fase

2	3	5	7	10	20	30	40	auto	log2	None	Maxfeat
0.469988	0.473317	0.47488	0.487268	0.489275	0.481394	0.496343	0.488664	0.487268	0.47488	0.493221	**2**
0.635274	0.638806	0.643696	0.640234	0.643198	0.645106	0.645714	0.648791	0.640234	0.643696	0.646259	**5**
0.627995	0.628529	0.634203	0.635826	0.632315	0.63573	0.636566	0.64128	0.635826	0.634203	0.636909	**10**
0.67432	0.674577	0.674503	0.673751	0.675437	0.670703	0.671148	0.673707	0.673751	0.674503	0.676742	**15**
0.682754	0.685136	0.684028	0.687722	0.688302	0.682697	0.686313	0.683131	0.687722	0.684028	0.682786	**25**
0.693946	0.68405	0.689523	0.687409	0.688989	0.689761	0.684976	0.684276	0.687409	0.689523	0.685464	**50**
0.701514	0.692348	0.699963	0.698225	0.69702	0.697727	0.693186	0.691643	0.698225	0.699963	0.69215	**75**
0.703731	0.694022	0.698695	0.697134	0.694927	0.697039	0.692518	0.691473	0.697134	0.698695	0.692664	**100**
0.706109	0.699002	0.70145	0.703931	0.701973	0.701498	0.695003	0.694863	0.703931	0.70145	0.694543	**125**
0.70433	0.698344	0.70102	0.699714	0.70003	0.69865	0.693704	0.695018	0.699714	0.70102	0.692267	**150**
0.707828	0.700906	0.702066	0.70176	0.70098	0.697443	0.697192	0.696885	0.70176	0.702066	0.691981	**200**
0.70753	0.701802	0.701851	0.702715	0.703322	0.700866	0.698502	0.696997	0.702715	0.701851	0.694573	**250**
											Nestimators

Tabla 22. Tabla donde se muestra los valores de F1-score para los parámetros de Random Forest

1	3	5	7	10	13	14	15	17	20	22	25	30	35	None	MAXDEPTH
0.7079	0.7086	0.7033	0.7051	0.7059	0.7061	0.7041	0.7065	0.706	0.706	0.704	0.7096	0.7059	0.7047	0.7078	**1**
0.7079	0.7087	0.7034	0.705	0.7039	0.7031	0.7067	0.7068	0.7052	0.7059	0.705	0.7077	0.7065	0.7072	0.7065	**3**
0.7079	0.7087	0.7026	0.7058	0.7041	0.7068	0.7071	0.7051	0.7065	0.7078	0.708	0.709	0.7084	0.7088	0.7088	**5**
0.7079	0.7088	0.704	0.7048	0.7066	0.7068	0.7081	0.7046	0.7069	0.7055	0.708	0.7076	0.7074	0.7072	0.707	**7**
0.7079	0.7088	0.7054	0.7029	0.7054	0.7049	0.7064	0.7081	0.7078	0.7064	0.708	0.7049	0.7055	0.7058	0.7064	**10**
0.7079	0.7087	0.7059	0.7054	0.7045	0.7062	0.7054	0.7061	0.706	0.709	0.708	0.7058	0.7067	0.7067	0.7067	**13**
0.7079	0.7086	0.7045	0.7056	0.7046	0.7054	0.7076	0.7055	0.7063	0.7057	0.705	0.7054	0.7061	0.7062	0.7062	**15**
0.7079	0.7086	0.703	0.7041	0.7044	0.7074	0.708	0.7062	0.707	0.7064	0.706	0.7073	0.707	0.707	0.707	**20**
0.7079	0.7082	0.7029	0.704	0.7034	0.7051	0.7076	0.7086	0.7073	0.7079	0.707	0.7061	0.706	0.706	0.706	**25**
0.7079	0.7083	0.7024	0.7036	0.7036	0.7048	0.7055	0.706	0.7055	0.705	0.705	0.7049	0.7054	0.7054	0.7054	**30**
0.7079	0.7083	0.7033	0.7043	0.7054	0.7053	0.7057	0.7052	0.705	0.7061	0.707	0.7055	0.7056	0.7056	0.7056	**35**
0.7079	0.7084	0.7039	0.7067	0.7036	0.7055	0.7035	0.7067	0.706	0.7066	0.706	0.7068	0.7068	0.7068	0.7068	**40**
0.7079	0.7081	0.7038	0.7032	0.7037	0.7058	0.7044	0.7035	0.7057	0.7043	0.706	0.7061	0.7061	0.7061	0.7061	**45**
0.7079	0.7083	0.7045	0.7046	0.7042	0.7056	0.7055	0.7073	0.7057	0.7064	0.706	0.7066	0.7066	0.7066	0.7066	**50**
0.7079	0.7088	0.7035	0.7035	0.7041	0.7056	0.7044	0.7052	0.7057	0.7048	0.704	0.7045	0.7045	0.7045	0.7045	**70**
															MINSAMPL

Tabla 23. Tabla donde se muestran los valores de F1-score en exp.1 en segunda fase

1	3	5	7	10	13	14	15	17	20	22	25	30	35	None	MAXDEPTH
0.7058	0.7097	0.7041	0.7039	0.706	0.707	0.7029	0.7078	0.7071	0.7055	0.7037	0.709	0.7069	0.7049	0.7075	**1**
0.7058	0.7097	0.7042	0.7039	0.7048	0.7045	0.7066	0.7059	0.706	0.7073	0.7048	0.7093	0.7078	0.7075	0.7075	**3**
0.7058	0.7097	0.7034	0.7052	0.7035	0.7066	0.7064	0.7063	0.7071	0.7088	0.7081	0.7086	0.7074	0.7106	0.7112	**5**
0.7058	0.7096	0.7046	0.7061	0.7084	0.7068	0.7082	0.7054	0.7066	0.7056	0.7078	0.7099	0.709	0.707	0.7075	**7**
0.7058	0.7097	0.7058	0.7017	0.7058	0.7051	0.7079	0.706	0.7071	0.7068	0.707	0.7062	0.7068	0.7076	0.7078	**10**
0.7058	0.7095	0.7057	0.7058	0.706	0.7052	0.7061	0.7081	0.7067	0.7087	0.7091	0.7085	0.7085	0.7084	0.7084	**13**
0.7058	0.7096	0.7056	0.7059	0.7052	0.7044	0.7071	0.706	0.707	0.7074	0.7063	0.7061	0.707	0.7064	0.7064	**15**
0.7058	0.7097	0.7043	0.7053	0.7032	0.7074	0.7089	0.7062	0.7084	0.7067	0.7061	0.7074	0.707	0.707	0.707	**20**
0.7058	0.7094	0.705	0.7048	0.7048	0.7064	0.7064	0.7102	0.7059	0.7073	0.707	0.7075	0.7074	0.7074	0.7074	**25**
0.7058	0.7094	0.7031	0.7047	0.7049	0.7043	0.7061	0.7061	0.7053	0.7053	0.7048	0.7064	0.706	0.706	0.706	**30**
0.7058	0.7095	0.7044	0.7033	0.7042	0.7047	0.7064	0.7049	0.706	0.707	0.7063	0.7054	0.7056	0.7056	0.7056	**35**
0.7058	0.7096	0.7052	0.7063	0.7033	0.7052	0.7055	0.7066	0.7063	0.7061	0.7064	0.7064	0.7064	0.7064	0.7064	**40**
0.7058	0.7097	0.7044	0.705	0.7034	0.7069	0.7047	0.705	0.7047	0.7048	0.7055	0.7049	0.7049	0.7049	0.7049	**45**
0.7058	0.7094	0.7054	0.7059	0.7041	0.7062	0.7046	0.7074	0.7065	0.7067	0.706	0.7064	0.7064	0.7064	0.7064	**50**
0.7058	0.7097	0.7051	0.7052	0.7046	0.7056	0.7041	0.7065	0.7059	0.7057	0.7058	0.7058	0.7058	0.7058	0.7058	**70**
															MINSAMPL

Tabla 24. Tabla donde se muestran los valores de F1-score en exp.2 en segunda fase

1	3	5	7	10	13	14	15	17	20	22	25	30	35	None	MAXDEPTH
0.7087	0.7072	0.7024	0.7034	0.7057	0.7042	0.7038	0.7058	0.7019	0.7036	0.7012	0.704	0.7046	0.7058	0.7061	**1**
0.7087	0.7072	0.7021	0.7033	0.7017	0.703	0.7053	0.7062	0.7048	0.7025	0.7033	0.7046	0.7062	0.7072	0.7083	**3**
0.7087	0.7072	0.7013	0.7036	0.7036	0.7061	0.7064	0.7035	0.7041	0.7072	0.7069	0.7074	0.7065	0.7088	0.7078	**5**
0.7087	0.7073	0.7015	0.7027	0.7069	0.704	0.7053	0.7058	0.7045	0.7045	0.7045	0.7051	0.7043	0.706	0.7058	**7**
0.7087	0.7073	0.7037	0.7009	0.7057	0.7046	0.705	0.7062	0.7086	0.7056	0.7078	0.7045	0.7056	0.7062	0.7062	**10**
0.7087	0.7072	0.704	0.7037	0.703	0.7041	0.7055	0.7054	0.7043	0.7083	0.7085	0.7046	0.7059	0.706	0.706	**13**
0.7087	0.7072	0.7032	0.7046	0.7044	0.7038	0.7074	0.7036	0.7048	0.7055	0.7028	0.7038	0.7035	0.7039	0.7039	**15**
0.7087	0.7069	0.702	0.7042	0.704	0.7068	0.7069	0.7045	0.7054	0.7061	0.7059	0.7056	0.706	0.706	0.706	**20**
0.7087	0.7063	0.7033	0.7036	0.7047	0.7059	0.7065	0.7073	0.7053	0.7073	0.7059	0.7053	0.7056	0.7056	0.7056	**25**
0.7087	0.7063	0.7022	0.7033	0.7047	0.7033	0.7061	0.7042	0.7035	0.7051	0.7048	0.7063	0.706	0.706	0.706	**30**
0.7087	0.7063	0.7018	0.7016	0.7061	0.7049	0.7064	0.704	0.7044	0.704	0.7043	0.7052	0.7054	0.7054	0.7054	**35**
0.7087	0.7063	0.7019	0.7041	0.7017	0.7054	0.7046	0.7051	0.7047	0.7064	0.7057	0.7062	0.7062	0.7062	0.7062	**40**
0.7087	0.7065	0.7013	0.7024	0.7014	0.7044	0.7023	0.7054	0.7041	0.7034	0.7034	0.7035	0.7035	0.7035	0.7035	**45**
0.7087	0.7065	0.7017	0.7045	0.704	0.7062	0.7052	0.7059	0.7074	0.7043	0.7055	0.7055	0.7055	0.7055	0.7055	**50**
0.7087	0.7071	0.7004	0.7022	0.7046	0.7047	0.7031	0.7045	0.7038	0.7033	0.7033	0.7033	0.7033	0.7033	0.7033	**70**
															MINSAMPL

Tabla 25. Tabla donde se muestran los valores de F1-score en exp.3 en segunda fase

1	3	5	7	10	13	14	15	17	20	22	25	30	35	None	MAXDEPTH
0.7078	0.7061	0.7037	0.7044	0.7049	0.7058	0.7029	0.7048	0.7049	0.7033	0.7025	0.7077	0.7048	0.7038	0.7043	**1**
0.7078	0.7062	0.7038	0.7029	0.7035	0.7031	0.7054	0.7074	0.7037	0.7038	0.7039	0.7058	0.7054	0.7071	0.709	**3**
0.7078	0.7062	0.7027	0.704	0.7033	0.7066	0.7066	0.7035	0.7062	0.7086	0.7059	0.7078	0.7063	0.709	0.7086	**5**
0.7078	0.7061	0.7025	0.7047	0.7058	0.7053	0.7067	0.7041	0.7059	0.7053	0.7072	0.7063	0.7058	0.7047	0.7048	**7**
0.7078	0.7061	0.7051	0.6999	0.7045	0.7057	0.7057	0.7068	0.7071	0.7075	0.7072	0.7048	0.7058	0.7052	0.7054	**10**
0.7078	0.706	0.7051	0.7037	0.7038	0.7056	0.7048	0.7047	0.7066	0.7075	0.7082	0.7069	0.7078	0.7076	0.7076	**13**
0.7078	0.706	0.7035	0.7047	0.7045	0.7045	0.708	0.7038	0.705	0.7065	0.7034	0.7043	0.7037	0.7034	0.7034	**15**
0.7078	0.7058	0.7027	0.7039	0.7037	0.7087	0.7078	0.706	0.7057	0.7062	0.7051	0.7073	0.7054	0.7054	0.7054	**20**
0.7078	0.7058	0.703	0.7024	0.7042	0.7052	0.7066	0.7072	0.706	0.7075	0.7049	0.7047	0.7058	0.7058	0.7058	**25**
0.7078	0.7059	0.7013	0.704	0.7043	0.7041	0.7056	0.7053	0.7041	0.7063	0.705	0.706	0.7064	0.7064	0.7064	**30**
0.7078	0.7062	0.7014	0.7017	0.7057	0.705	0.706	0.7035	0.7043	0.7055	0.706	0.7067	0.7064	0.7064	0.7064	**35**
0.7078	0.706	0.7024	0.7036	0.7025	0.7059	0.7047	0.707	0.7047	0.7062	0.7051	0.7053	0.7053	0.7053	0.7053	**40**
0.7078	0.7059	0.7031	0.7012	0.7025	0.7058	0.7024	0.7056	0.7058	0.7038	0.7045	0.7047	0.7047	0.7047	0.7047	**45**
0.7078	0.7059	0.7033	0.7035	0.7038	0.7052	0.7061	0.7061	0.7064	0.7045	0.7059	0.7059	0.7059	0.7059	0.7059	**50**
0.7078	0.7059	0.7028	0.7019	0.7024	0.7054	0.704	0.7042	0.7033	0.7036	0.7036	0.7036	0.7036	0.7036	0.7036	**70**
															MINSAMPL

Tabla 26. Tabla donde se muestran los valores de F1-score en exp.4 en segunda fase

1	3	5	7	10	13	14	15	17	20	22	25	30	35	None	MAXDEPTH
0.6982	0.691	0.6954	0.6972	0.6994	0.7001	0.701	0.7033	0.7054	0.7004	0.7026	0.7039	0.7007	0.7046	0.7039	**1**
0.6982	0.691	0.695	0.6979	0.6977	0.7035	0.7023	0.7018	0.6997	0.6983	0.7049	0.6986	0.7029	0.699	0.6988	**3**
0.6982	0.691	0.6941	0.6971	0.7008	0.7	0.7013	0.7046	0.703	0.6998	0.7028	0.7037	0.7045	0.7034	0.7027	**5**
0.6982	0.691	0.6953	0.6978	0.6983	0.7043	0.7024	0.7013	0.7042	0.7023	0.7027	0.7029	0.7016	0.7011	0.7012	**7**
0.6982	0.691	0.6952	0.6982	0.6998	0.7022	0.7	0.7018	0.7034	0.7031	0.7006	0.7017	0.7027	0.7024	0.7024	**10**
0.6982	0.6909	0.6955	0.6968	0.6982	0.7012	0.7023	0.704	0.7004	0.703	0.7019	0.7035	0.7025	0.7027	0.7029	**13**
0.6982	0.6909	0.6945	0.6967	0.6998	0.702	0.7023	0.702	0.7047	0.7013	0.7027	0.7031	0.7038	0.7039	0.7039	**15**
0.6982	0.6909	0.6952	0.6986	0.7009	0.7014	0.7	0.6987	0.7027	0.7032	0.7028	0.7006	0.702	0.702	0.702	**20**
0.6982	0.6908	0.6952	0.6977	0.701	0.7025	0.7007	0.7025	0.7016	0.7011	0.7002	0.7013	0.7013	0.7013	0.7013	**25**
0.6982	0.6907	0.6947	0.6974	0.7008	0.7025	0.7015	0.7008	0.7006	0.7017	0.702	0.7008	0.7011	0.7011	0.7011	**30**
0.6982	0.6907	0.6952	0.6962	0.6977	0.7007	0.7	0.7016	0.7009	0.7002	0.7009	0.7009	0.7009	0.7009	0.7009	**35**
0.6982	0.6907	0.6948	0.6975	0.7001	0.7003	0.7003	0.7004	0.6992	0.7006	0.7006	0.6999	0.6999	0.6999	0.6999	**40**
0.6982	0.6909	0.6947	0.6979	0.6999	0.7018	0.7007	0.6999	0.7034	0.7013	0.7023	0.7019	0.7019	0.7019	0.7019	**45**
0.6982	0.6906	0.6946	0.6973	0.7005	0.6985	0.7009	0.7003	0.7024	0.702	0.7016	0.7016	0.7016	0.7016	0.7016	**50**
0.6982	0.6904	0.6956	0.6962	0.7002	0.6999	0.6999	0.7006	0.7015	0.7018	0.7019	0.7019	0.7019	0.7019	0.7019	**70**
															MINSAMPL

Tabla 27. Tabla donde se muestran los valores de F1-score en exp.5 en segunda fase

1	3	5	7	10	13	14	15	17	20	22	25	30	35	None	MAXDEPTH
0.7091	0.7035	0.7009	0.7016	0.7028	0.7017	0.7038	0.7048	0.7	0.7025	0.6995	0.7034	0.7036	0.7004	0.7037	**1**
0.7091	0.7035	0.701	0.7015	0.7024	0.7015	0.7051	0.7038	0.702	0.7034	0.7042	0.7038	0.7051	0.7057	0.7076	**3**
0.7091	0.7035	0.7012	0.7036	0.7042	0.7077	0.7032	0.7037	0.7041	0.7055	0.7045	0.7048	0.7044	0.7077	0.7068	**5**
0.7091	0.7036	0.7021	0.7019	0.7064	0.7033	0.7072	0.7039	0.7036	0.7038	0.7044	0.7049	0.7036	0.7059	0.705	**7**
0.7091	0.7036	0.7033	0.7001	0.7051	0.7051	0.7067	0.7085	0.7064	0.7044	0.7071	0.702	0.7049	0.7041	0.7041	**10**
0.7091	0.7036	0.7033	0.7046	0.7013	0.7046	0.7044	0.7031	0.7053	0.7072	0.7074	0.7048	0.7063	0.7061	0.7061	**13**
0.7091	0.7036	0.7019	0.7046	0.7042	0.7041	0.7069	0.7041	0.7048	0.702	0.7017	0.7038	0.7018	0.7017	0.7017	**15**
0.7091	0.7037	0.7023	0.7028	0.7035	0.7037	0.7073	0.7036	0.7039	0.7067	0.7058	0.7062	0.7068	0.7068	0.7068	**20**
0.7091	0.7036	0.7043	0.7038	0.7041	0.7059	0.7047	0.7052	0.7062	0.7066	0.706	0.7044	0.7042	0.7042	0.7042	**25**
0.7091	0.7037	0.7022	0.7043	0.7048	0.7018	0.7052	0.7038	0.7042	0.7049	0.7046	0.7056	0.7051	0.7051	0.7051	**30**
0.7091	0.7037	0.7007	0.7028	0.7064	0.705	0.7063	0.7034	0.7033	0.7026	0.7045	0.7044	0.7044	0.7044	0.7044	**35**
0.7091	0.7036	0.7003	0.7026	0.7004	0.7051	0.7044	0.7064	0.7043	0.7058	0.7053	0.7047	0.7047	0.7047	0.7047	**40**
0.7091	0.7035	0.7005	0.7023	0.7028	0.7031	0.7025	0.7031	0.7039	0.7036	0.7033	0.7039	0.7039	0.7039	0.7039	**45**
0.7091	0.7036	0.7009	0.7042	0.7024	0.7037	0.7042	0.7049	0.7036	0.7012	0.7026	0.7026	0.7026	0.7026	0.7026	**50**
0.7091	0.7037	0.701	0.701	0.7041	0.704	0.7026	0.7031	0.7034	0.7024	0.7024	0.7024	0.7024	0.7024	0.7024	**70**
															MINSAMPL

Tabla 28. Tabla donde se muestran los valores de F1-score en exp.6 en segunda fase

1	3	5	7	10	13	14	15	17	20	22	25	30	35	None	MAXDEPTH
0.688	0.685	0.6922	0.6951	0.7009	0.701	0.702	0.7018	0.7033	0.7027	0.7025	0.7033	0.703	0.7025	0.7033	**1**
0.688	0.685	0.6923	0.6967	0.6991	0.7023	0.7039	0.704	0.7049	0.703	0.706	0.7022	0.7057	0.7038	0.7035	**3**
0.688	0.685	0.6921	0.6962	0.6981	0.7041	0.7033	0.7052	0.7016	0.7021	0.7045	0.7032	0.7028	0.7042	0.7035	**5**
0.688	0.685	0.6921	0.6933	0.7004	0.7024	0.7041	0.7021	0.7031	0.7034	0.704	0.7058	0.7047	0.7071	0.7071	**7**
0.688	0.685	0.6932	0.6955	0.701	0.7049	0.7048	0.7038	0.7055	0.7036	0.7042	0.7026	0.7027	0.7018	0.7024	**10**
0.688	0.685	0.6928	0.6941	0.6983	0.7029	0.703	0.7037	0.7022	0.7013	0.7015	0.7036	0.7027	0.7024	0.7024	**13**
0.688	0.685	0.6923	0.6939	0.7002	0.7039	0.7015	0.7021	0.7021	0.703	0.7028	0.7035	0.7036	0.7041	0.7041	**15**
0.688	0.685	0.6923	0.6948	0.7009	0.7047	0.7038	0.7022	0.7038	0.7041	0.7033	0.7053	0.705	0.705	0.705	**20**
0.688	0.684	0.6934	0.6938	0.6998	0.7014	0.7032	0.7032	0.7028	0.7025	0.7038	0.7043	0.7042	0.7042	0.7042	**25**
0.688	0.684	0.6932	0.6955	0.6995	0.7021	0.7012	0.7006	0.7011	0.7031	0.7024	0.7034	0.7034	0.7034	0.7034	**30**
0.688	0.684	0.6936	0.6949	0.6983	0.7025	0.7011	0.7013	0.7018	0.7009	0.7009	0.701	0.701	0.701	0.701	**35**
0.688	0.685	0.6933	0.6936	0.6974	0.7018	0.7017	0.6998	0.7012	0.7007	0.7002	0.7004	0.7004	0.7004	0.7004	**40**
0.688	0.685	0.6937	0.6925	0.6992	0.7006	0.7004	0.7006	0.7013	0.7002	0.7009	0.7008	0.7008	0.7008	0.7008	**45**
0.688	0.685	0.6927	0.6934	0.6986	0.7014	0.6989	0.702	0.701	0.7002	0.7002	0.7002	0.7002	0.7002	0.7002	**50**
0.688	0.684	0.6911	0.6948	0.6997	0.6991	0.6995	0.7012	0.7002	0.7001	0.7001	0.7001	0.7001	0.7001	0.7001	**70**
															MINSAMPL

Tabla 29. Tabla donde se muestran los valores de F1-score en exp.7 en segunda fase

A2.Código de programación

Preparación y transformación de los datos

```
import numpy as np

import pandas as pd

dades=pd.read_csv("OnlineNewsPopularity.csv")

dades.columns= ["url","timedelta","n_tokens_title","n_tokens_content", "n_unique_tokens",
"n_non_stop_words","n_non_stop_unique_tokens","num_hrefs","num_self_hrefs","num_i
mgs","num_videos","average_token_length","num_keywords", "data_channel_is_lifestyle",
"data_channel_is_entertainment","data_channel_is_bus","data_channel_is_socmed",
"data_channel_is_tech","data_channel_is_world","kw_min_min","kw_max_min",
"kw_avg_min","kw_min_max","kw_max_max","kw_avg_max","kw_min_avg","kw_max_av
g","kw_avg_avg","self_reference_min_shares","self_reference_max_shares",
"self_reference_avg_shares","weekday_is_monday","weekday_is_tuesday",
"weekday_is_wednesday","weekday_is_thursday","weekday_is_friday","weekday_is_satu
rday", "weekday_is_sunday","is_weekend", "LDA_00", "LDA_01", "LDA_02", "LDA_03",
"LDA_04", "global_subjectivity", "global_sentiment_polarity", "global_rate_positive_words",
"global_rate_negative_words","rate_positive_words","rate_negative_words",
"avg_positive_polarity","min_positive_polarity","max_positive_polarity",
"avg_negative_polarity","min_negative_polarity","max_negative_polarity",
"title_subjectivity","title_sentiment_polarity","abs_title_subjectivity",
"abs_title_sentiment_polarity", "shares"]

del dades['url']

del dades['timedelta']

dades2=dades.copy()
```

```
def categoria3(j):
    df=j.copy()
    categorianova=[]
    for e in df["shares"]:
        if e >=2700:
            categorianova.append("alt")
        elif e <=950:
            categorianova.append("baix")
        else:
            categorianova.append("mig")
    df["sharescatg"]=categorianova
    label=df["sharescatg"]
    del df["shares"]
    del df["sharescatg"]
    return label,df

def categoria2(j):
    df=j.copy()
    df["sharescatg2"]=(df["shares"]>=1400).astype(int)
    label=df["sharescatg2"]
    del df["sharescatg2"]
    del df["shares"]
    return label,df

def añadircolumnanueva(x, columna1,columna2,ncolumna):
    df=x.copy()
df[str(ncolumna)]=2*(df[str(columna1)]*df[str(columna2)])/(df[str(columna1)]+df[str(column
a2)])
    df=df.replace([np.inf, -np.inf], np.nan)
    df[str(ncolumna)]=df[str(ncolumna)].fillna(0)
```

```
def categoria32(j):
    df=j.copy()
    categorianova=[]
    for e in df["shares"]:
        if e >=2100:
            categorianova.append("alt")
        elif e <=1099:
            categorianova.append("baix")
        else:
            categorianova.append("mig")
    df["sharescatg"]=categorianova
    label=df["sharescatg"]
    del df["shares"]
    del df["sharescatg"]
    return label,df


def holdout(x,label):
    from sklearn.model_selection import train_test_split(X_train,X_test,Y_train,Y_test
=train_test_split(x,label,test_size=0.3,random_state=17,stratify=label)
    return X_train, X_test,Y_train,Y_test
```

Regression Logística

```
from sklearn import linear_model
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
def logisticadef(X_train,Y_train,X_test,Y_test):
    modelo=linear_model.LogisticRegression(random_state=17)
    modelo.fit(X_train,Y_train)
    prediccion=modelo.predict(X_test)
    print(accuracy_score(Y_test,prediccion))
    confusion=confusion_matrix(Y_test, prediccion)
    print(classification_report(Y_test, prediccion))
    recall=confusion[1][1]/(confusion[1][1]+confusion[1][0])
    precisio=confusion[1][1]/(confusion[0][1]+confusion[1][1])
```

```
    F1=(2*recall*precisio)/(recall+precisio)
    print(F1)
    return accuracy_score(Y_test, prediccion), confusion_matrix(Y_test,prediccion),
classification_report(Y_test, prediccion)
```

Random Forest

```
def randomforestmodeldefault(x_train,label_train,x_test,label_test):
    modelo = RandomForestClassifier(random_state=17)
    modelo.fit(x_train,label_train)
    prediccion=modelo.predict(x_test)
    pyplot.bar(range(len(modelo.feature_importances_)), modelo.feature_importances_)
    pyplot.show()
    confusion=confusion_matrix(label_test,prediccion)
    recall=confusion[1][1]/(confusion[1][1]+confusion[1][0])
    precisio=confusion[1][1]/(confusion[0][1]+confusion[1][1])
    F1=(2*recall*precisio)/(recall+precisio)
    print(F1)
    return accuracy_score(label_test,prediccion),confusion_matrix(label_test,prediccion),
classification_report(label_test,prediccion)

 def randomforestmodel(x_train,label_train,x_test,label_test,min_simp,max_d):
modelo=RandomForestClassifier(max_depth=max_d,criterion='entropy',min_samples_leaf
=min_simp,random_state=17)
    modelo.fit(x_train, label_train)
    prediccion=modelo.predict(x_test)
    return accuracy_score(label_test,prediccion),confusion_matrix(label_test,prediccion),
classification_report(label_test,prediccion)

def randomforestmodel2(x_train,label_train,x_test,label_test,n_est,max_d):
modelo=RandomForestClassifier(max_depth=13,max_features=max_d,criterion='entropy',
n_estimators=n_est,random_state=17,min_samples_leaf=45)
    modelo.fit(x_train, label_train)
    prediccion=modelo.predict(x_test)
    return accuracy_score(label_test,prediccion),confusion_matrix(label_test,prediccion),
classification_report(label_test,prediccion)
```

```
def preci1(x_train,label_train,x_test,label_test):
    precision=[]
    num=0
    for e in [1,3,5,7,10,13,15,20,25,30,35,40,45,50,70]:
        lista=[]
        for i in [1,3,5,7,10,13,14,15,17,20,22,25,30,35,None]:

accuracy,confusion,clas=randomforestmodel(x_train,label_train,x_test,label_test,e,i)
            recall=confusion[1][1]/(confusion[1][1]+confusion[1][0])
            precisio=confusion[1][1]/(confusion[0][1]+confusion[1][1])
            F1=(2*recall*precisio)/(recall+precisio)
            lista.append(F1)
            num=num+1
        precision.append(lista)
    return precision

def preci2(x_train,label_train,x_test,label_test):
    precision=[]
    num=0
    for e in [2,5,10,15,25,50,75,85,100,125,150,200,250]:
        lista=[]
        for i in [2,3,5,7,10,20,30,40,'auto','log2',None]:

accuracy,confusion,clas=randomforestmodel2(x_train,label_train,x_test,label_test,e,i)
            recall=confusion[1][1]/(confusion[1][1]+confusion[1][0])
            precisio=confusion[1][1]/(confusion[0][1]+confusion[1][1])
            F1=(2*recall*precisio)/(recall+precisio)
            lista.append(F1)
            num=num+1
        precision.append(lista)
    return precision
```

```
def conjuntodatos2(ped):
    n_estimators=[2,5,10,15,25,50,75,85,100,125,150,200,250]
    max_features=[2,3,5,7,10,20,30,40,'auto','log2',None]
    treedef=pd.DataFrame(columns=max_features,index=n_estimators)
    for idx in range(len(n_estimators)):
        for ind in range(len(max_features)):
            treedef.at[n_estimators[idx],max_features[ind]]=ped[idx][ind]
    writer = ExcelWriter('Prueba6_13-45.xlsx')
    treedef.to_excel(writer, 'Hoja de datos', index=False)
    writer.save()
    return treedef

def conjuntodatos1(ped):
    min_samples_leaf=[1,3,5,7,10,13,15,20,25,30,35,40,45,50,70]
    max_depth=[1,3,5,7,10,13,14,15,17,20,22,25,30,35,None]
    treedef=pd.DataFrame(columns=max_depth,index=min_samples_leaf)
    for idx in range(len(min_samples_leaf)):
        for ind in range(len(max_depth)):
            treedef.at[min_samples_leaf[idx],max_depth[ind]]=ped[idx][ind]
    writer = ExcelWriter('minsamplmaxdepth.xlsx')
    treedef.to_excel(writer, 'Hoja de datos', index=False)
    writer.save()
    return treedef

def nuevorandomforestmodel23(x_train,label_train,x_test,label_test,min_simp,max_d):
modelo=RandomForestClassifier(max_features=2,n_estimators=100,max_depth=max_d,
criterion='entropy',min_samples_leaf=min_simp,random_state=17)
    modelo.fit(x_train, label_train)
    prediccion=modelo.predict(x_test)
    return accuracy_score(label_test,prediccion),confusion_matrix(label_test,prediccion),
classification_report(label_test,prediccion)
```

```
def nuevorandomforestmodel2(x_train,label_train,x_test,label_test,n_est,max_d):
modelo=RandomForestClassifier(max_features=max_d,criterion='entropy',n_estimators=n
_est,random_state=17)
    modelo.fit(x_train, label_train)
    prediccion=modelo.predict(x_test)
    return     accuracy_score(label_test,prediccion),confusion_matrix(label_test,prediccion),
classification_report(label_test,prediccion)


def preci12(x_train,label_train,x_test,label_test):
    precision=[]
    num=0
    for e in [1,3,5,7,10,13,15,20,25,30,35,40,45,50,70]:
        lista=[]
        for i in [1,3,5,7,10,13,14,15,17,20,22,25,30,35,None]:
   accuracy,confusion,clas=nuevorandomforestmodel23
(x_train,label_train,x_test,label_test,e,i)
            recall=confusion[1][1]/(confusion[1][1]+confusion[1][0])
            precisio=confusion[1][1]/(confusion[0][1]+confusion[1][1])
            F1=(2*recall*precisio)/(recall+precisio)
            lista.append(F1)
            print(num)
            num=num+1
        precision.append(lista)
    return precision


def conjuntodatos12(ped):
    min_samples_leaf=[1,3,5,7,10,13,15,20,25,30,35,40,45,50,70]
    max_depth=[1,3,5,7,10,13,14,15,17,20,22,25,30,35,None]
    treedef=pd.DataFrame(columns=max_depth,index=min_samples_leaf)
    for idx in range(len(min_samples_leaf)):
        for ind in range(len(max_depth)):
            treedef.at[min_samples_leaf[idx],max_depth[ind]]=ped[idx][ind]
    writer = ExcelWriter('Probatipo2_7_nest100-maxfeat2.xlsx')
    treedef.to_excel(writer, 'Hoja de datos', index=False)
    writer.save()
    return treedef
```

```
def preci22(x_train,label_train,x_test,label_test):
    precision=[]
    num=0
    for e in [2,5,10,15,25,50,75,85,100,125,150,200,250,500,700,1000]:
        lista=[]
        for i in [2,3,5,7,10,20,30,40,'auto','log2',None]:

accuracy,confusion,clas=nuevorandomforestmodel2(x_train,label_train,x_test,label_test,e,
i)
            recall=confusion[1][1]/(confusion[1][1]+confusion[1][0])
            precisio=confusion[1][1]/(confusion[0][1]+confusion[1][1])
            F1=(2*recall*precisio)/(recall+precisio)
            lista.append(F1)
            print(num)
            num=num+1
        precision.append(lista)
    return precision


def conjuntodatos22(ped):
    n_estimators=[2,5,10,15,25,50,75,85,100,125,150,200,250,500,700,1000]
    max_features=[2,3,5,7,10,20,30,40,'auto','log2',None]
    treedef=pd.DataFrame(columns=max_features,index=n_estimators)
    for idx in range(len(n_estimators)):
        for ind in range(len(max_features)):
            treedef.at[n_estimators[idx],max_features[ind]]=ped[idx][ind]
    writer = ExcelWriter('nestimmaxfeat4.xlsx')
    treedef.to_excel(writer, 'Hoja de datos', index=False)
    writer.save()
    return treedef
```